2018

Evaluation of Machine Learning Techniques for Early Identification of At-Risk Students

Mansour Hamoud Awaji
Nova Southeastern University, mawaji@live.com

Evaluation of Machine Learning Techniques for Early Identification of At-Risk Students

by
Mansour Awaji

A Dissertation Submitted in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
in
Information Systems

College of Engineering and Computing
Nova Southeastern University
2018

We hereby certify that this dissertation, submitted by Mansour Hamoud M. Awaji, conforms to acceptable standards and is fully adequate in scope and quality to fulfill the dissertation requirements for the degree of Doctor of Philosophy.

Sumitra Mukherjee, Ph.D.
Chairperson of Dissertation Committee

Nov 13, 2018
Date

Ling Wang, Ph.D.
Dissertation Committee Member

11/13/2018
Date

Michael J. Laszlo, Ph.D.
Dissertation Committee Member

11/13/2018
Date

Approved:

Meline Kevorkian, Ed.D.
Interim Dean, College of Engineering and Computing

11/13/18
Date

College of Engineering and Computing
Nova Southeastern University

2018

Abstract

An Abstract of a Dissertation Submitted to Nova Southeastern University
in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

Evaluation of Machine Learning Techniques for Early Identification of At-Risk Students

by

Mansour Awaji

October 2018

Student attrition is one of the long-standing problems facing higher education institutions despite the extensive research that has been undertaken to address it. To increase students' success and retention rates, there is a need for early alert systems that facilitate the identification of at-risk students so that remedial measures may be taken in time to reduce the risk. However, incorporating ML predictive models into early warning systems face two main challenges: improving the accuracy of timely predictions and the generalizability of predictive models across on-campus and online courses. The goal of this study was to develop and evaluate predictive models that can be applied to on-campus and online courses to predict at-risk students based on data collected from different stages of a course: start of the course, 4th week, 8th week, and 12th week.

In this research, several supervised machine learning algorithms were trained and evaluated on their performance. This study compared the performance of single classifiers (Logistic Regression, Decision Trees, Naïve Bayes, and Artificial Neural Networks) and ensemble classifiers (using bagging and boosting techniques). Their performance was evaluated in term of sensitivity, specificity, and Area Under Curve (AUC). A total of four experiments were conducted based on data collected from different stages of the course. In the first experiment, the classification algorithms were trained and evaluated based on data collected before the beginning of the semester. In the second experiment, the classification algorithms were trained and evaluated based on week-four data. Similarly, in the third and fourth experiments, the classification algorithms were trained and evaluated based on week-eight and week-12 data.

The results demonstrated that ensemble classifiers were able to achieve the highest classification performance in all experiments. Additionally, the results of the generalizability analysis showed that the predictive models were able to attain a similar performance when used to classify on-campus and online students. Moreover, the Extreme Gradient Boosting (XGBoost) classifier was found to be the best performing classifier suited for the at-risk students' classification problem and was able to achieve an

AUC of ≈ 0.89, a sensitivity of ≈ 0.81, and specificity of ≈ 0.81 using data available at the start of a course. Finally, the XGBoost classifier was able to improve by 1% for each subsequent four weeks dataset reaching an AUC of ≈ 0.92, a sensitivity of ≈ 0.84, and specificity of ≈ 0.84 by week 12. While the additional learning management system's (LMS) data helped in improving the prediction accuracy consistently as the course progresses, the improvement was marginal. Such findings suggest that the predictive models can be used to identify at-risk students even in courses that do not make significant use of LMS.

The results of this research demonstrated the usefulness and effectiveness of ML techniques for early identification of at-risk students. Interestingly, it was found that fairly reliable predictions can be made at the start of the semester, which is significant in that help can be provided to at-risk students even before the course starts. Finally, it is hoped that the results of this study advance the understanding of the appropriateness and effectiveness of ML techniques when used for early identification of at-risk students.

Acknowledgments

First and foremost, I would like to express my gratitude to my parents, wife, brothers, and sisters for their endless support, encouragement, and patience throughout this endeavor.

I would also like to thank my advisor, Dr. Sumitra Mukherjee, for his support, guidance, and assistance throughout this dissertation process. Dr. Mukherjee was always available to answer my questions and point me in right direction. I would also like to thank Dr. Ling Wang and Dr. Michael Laszlo, the dissertation committee members, for their guidance, support, and helpful suggestions.

Table of Contents

List of Tables

Tables

List of Figures

Figures

Chapter 1

Introduction

Background

For decades, student retention has been a serious problem for higher education institutions around the world, and a great deal of research has been undertaken to investigate the factors that contribute to this issue. Student retention rates correspond to the percentage of students who were admitted to an institution and were able to successfully graduate within six years of their start date (Tinto, 2012). In general, not every admitted student is able to complete his or her degree. Some students fail to finish their degrees because of academic challenges or other external factors. Student attrition results in negative consequences for students, families, the institutions, and the economy. Students lose future career and potential income when they leave the university before completing their degree. Higher education institutions lose potential revenue and alumni donations. Moreover, in the United States, it is estimated that student attrition costs higher education institutions and the government more than four billion dollars per year (Schneider & Yin, 2011; Seidman, 2012; Tinto, 1987; Tinto, 2012).

Despite the high cost of student attrition and the continued efforts from higher education institutions around the world to improve student retention, student attrition rates at higher education institutions remain relatively high across universities around the world. A report published by the National Center for Education Statistics revealed that, in

the United States, the average retention rate for public and private four-year institutions in 2015 was about 59% (Ginder, Kelly-Reid, & Mann, 2016).

A great deal of research has been devoted to investigate factors that affect student retention in higher education. Factors identified in the literature can be grouped into individual, institutional, and social factors. Of the various factors contributing to students' decisions to drop out, not surprisingly, academic success was found to be a significant factor in students' retention (DesJardins, Ahlburg, & McCall, 1999; Pascarella & Terenzini, 2005). Researchers concluded that an increase in academic achievement reduces the risk of student withdrawals (Murtaugh, Burns, & Schuster, 1999). A study conducted by Niemi and Gitin (2012) revealed that poor student performance at the course level is a significant predictor of student attrition. Moreover, Harvey and Luckman (2014) found that course failure was the strongest predictor of student attrition. Therefore, increasing student success at the course level would increase the likelihood of increasing student retention.

To promote student success, researchers claim that early intervention for struggling students has a positive impact on enhancing the students' academic results (Lizzio & Wilson, 2013). Zhang, Fei, Quddus, and Davis (2014) assert that early intervention programs that identify at-risk students and provide guidance to them helped to increase student retention. Early intervention programs for at-risk students should provide instructors with tools that help to identify at-risk students so that they can apply appropriate measures to improve the students learning process, which could lead to better outcomes. The first step in early intervention programs is the identification of at-risk students. Typically, instructors manually track the performance of their students and rely

on their own intuition to determine which of their students are at-risk of failing a course (Dietz-Uhler & Hurn, 2013). However, it is a difficult job for instructors to keep track of the performance of each student individually. Students who did not receive enough support when they need it may end up failing classes when they could have passed if early intervention measures had been in place (Meier, Xu, Atan, & van der Schaar, 2016). Consequently, a number of higher education institutions have taken the initiative to develop early warning systems that track and identify students who are struggling at the course level (Gašević, Dawson, Rogers, & Gasevic, 2016; Jayaprakash, Moody, Lauria, Regan, & Baron, 2014). An early alert system can facilitate the identification of at-risk students or predict student performance by analyzing students' data including demographics, academics, and learning management systems (LMS) usage data (Hu & Shih, 2014).

To mitigate the difficulties of identifying at-risk students, machine learning (ML) techniques have been applied by researchers to help in identifying struggling students (Peña-Ayala, 2014). It has been shown that it is feasible to predict students' success in a course and identify those who are at-risk using predictive models (Arnold & Pistilli, 2012; Macfadyen & Dawson, 2010). Initial works that applied ML techniques focused on developing predictive models that established how well students' performance can be predicted by applying different types of ML techniques to available student data (Hu & Shih, 2014; Romero & Ventura, 2013). While the results of the previous studies are promising, incorporating ML predictive models into early warning systems still face many challenges including the accuracy of timely predictions and the generalizability of predictive models. The majority of the developed predictive models were based on data

collected at the end of the semester. Predictions obtained by the end of semester come at a point of time where timely interventions may not be possible. Moreover, most of the studies that utilized ML techniques to predict students' performance were based on data obtained from a single or limited number of courses in a particular field of study, which led the researchers to question the generalizability of the predictive models across different student populations, courses, and disciplines (Gašević et al., 2016; Jayaprakash et al., 2014; Romero & Ventura, 2013).

Problem Statement

This research addressed the challenges associated with incorporating predictive models into early warning systems, namely the accuracy of timely predictions and the generalizability of predictive models across on-campus and online courses.

Most of the previous studies focused on the development of predictive models to predict students' performance based on data collected at the end of semester when a course has concluded and disregarded the value of early alert systems that need to identify struggling students while a course is still in progress (Hu & Shih, 2014). Timely prediction of at-risk students has been recognized as a critical factor for successful interventions meant to help struggling students. Providing early feedback to the students about their performance in a course gives them the opportunity to change what might be ineffective in their learning strategies. Many students start to recognize that they may be at-risk after receiving a poor midterm grade, which might be, for some students, too late to improve. The availability of detailed students' data makes it feasible for using ML techniques to predict at-risk students within a few weeks of a course start date. Such

predictions provide instructors and students with the opportunity to take remedial measures with enough time for improvement (Jayaprakash et al., 2014). While it is feasible to provide early predictions of at-risk students, achieving accurate timely predictions is challenging for various reasons. First, timely predictions of at-risk students must be obtained early enough in the semester to allow for timely intervention within the current term. The short window of time for intervention requires real-time analysis of students' data (Sander, 2016). Second, as most of the students are motivated at the beginning of the course, their scores and activities in earlier weeks may not be indicative of their scores and activities in later weeks and their overall performance. Third, timely predictions may vary in time from one student to another due to the variations of students' backgrounds and characteristics (Meier et al., 2016). Thus, this study explored whether data obtained from the first weeks of a course could provide an accurate prediction of at-risk students. In this research effort, students' data was obtained at four points of time (e.g.,1st week, 4th week, 8th week, and 12th week) during a 16-week semester.

An effective approach to improve the performance of early prediction models is to utilize ensemble techniques. Ensemble classifiers combine the outputs of multiple classification algorithms to classify new instances. Researchers from diverse disciplines have explored and applied ensemble methods in a broad variety of domains. It has been found that ensemble techniques constantly achieve better predictive performance than their base classifiers (Dietterich, 2000; Rokach, 2010). Therefore, in this research endeavor, a variety of ensemble classifiers were trained and evaluated to classify at-risk students.

Most of the studies that utilized ML techniques to predict students' performance were based on data obtained from a single or limited number of courses within a particular field of study (Gašević et al., 2016; Jayaprakash et al, 2014; Romero & Ventura, 2013). According to Romero, López, Luna, and Ventura (2013), factors that influence academic performance are diverse. They differ from one academic environment to another, from a subset of student populations to another, and from one cultural background to another. Predictive models that were developed based on a small sample size of courses within similar disciplines may not be applicable to other courses across the institution due to the differences in student populations, course design, and disciplines. Accordingly, there is a need to investigate if such predictive models are portable and provide similar prediction accuracy when applied to other courses with dissimilar student populations and contexts (Gašević et al., 2016; Jayaprakash et al, 2014; Romero & Ventura, 2013). As an initial step, this study investigated the generalizability of predictive models across courses offered for on-campus and online students. On-campus and online students differ in their characteristics. Online students are usually older than on-campus students. Moreover, online students generally enroll in the university as part-time students while they are full-time employees. Additionally, online students reportedly differ in their learning methods as online courses require high levels of self-motivation, self-regulation, and self-discipline (Bork & Rucks-Ahidiana, 2013; Kahu, Stephens, Leach, & Zepke, 2013; Quinn & Stein 2013).

This research effort explored whether data obtained from the first weeks of a course could provide an accurate prediction of at-risk students, and whether the developed predictive model worked satisfactorily for both on-campus and online courses.

Dissertation Goal

The goal of this study was to develop and evaluate predictive models that can be applied to on-campus and online courses to predict at-risk students based on data collected from different stages of a course: start of the course, 4th week, 8th week, and 12th week. The prediction accuracy of the developed predictive models was measured by true positive rates, true negative rates, and AUC. For the purpose of this research, the prediction task was formulated as a binary classification task to determine whether students are in good standing or at-risk. The target attribute that was employed to classify students as in good standing and at-risk was based on students' grades in a particular course where students with a grade worse than "C" were considered at-risk.

Research Questions

This study answered the following questions:

1- What are the best ML techniques to predict at-risk students?
2- Can acceptable accuracy be obtained by using ML techniques based on data collected from the first weeks of a course?
3- Can ensemble techniques improve the prediction accuracy of the base classifiers?
4- How does the prediction accuracy improve as the course progresses?
5- Can the predictive models achieve similar classification performance across on-campus and online courses?
6- What attributes are the best predictors?

Relevance and Significance

Student attrition is one of the long-standing problems facing higher education institutions (Tinto, 1987; Tinto, 2012). Despite the amount of research that has been undertaken to address student retention in higher education institutions, the graduation rates are still disturbingly low. A report published by ACT (2015) reported that since 2000, retention rates have consistently remained at approximately 50-53% among all four-year institutions. Moreover, in 2015, about 26% of all students enrolled in a four-year institution did not return for their second year (ACT, 2015). Students, higher education institutions, and the economy incur a high cost when students enrolled in four-year programs drop out (Tinto, 2012). The U.S. Bureau of Statistics (2016) reported that persons with a bachelor's degree could earn on average $24,000 more per year than individuals with just a high school diploma. Moreover, the unemployment rate for individuals with just a high school diploma is almost double the unemployment rate for those with a bachelor's degree. At the institutional level, higher education institutions lose potential revenue and alumni donations. Additionally, retention rates are considered as significant indicators of institutional quality and commitment to undergraduate education (Seidman, 2012; Tinto, 2012). Furthermore, in the United States, it is estimated that student attrition costs higher education institutions and the government more than four billion dollars per year (Schneider & Yin, 2011).

Academic success was found to be a significant factor in improving students' retention rates (DesJardins et al., 1999; Pascarella & Terenzini, 2005). Researchers concluded that an increase in academic achievement reduces the risk of student drop out (Murtaugh, Burns, & Schuster, 1999). Early intervention methods that are based on the

early identification of students who might be at-risk of failing a course have been called for to increase students' success and retention rates. The early identification of at-risk students provides instructors with the opportunity to help at-risk students succeed in the courses they are struggling with by providing them with the necessary resources and feedback (Hu & Shih, 2014; Zhang et al., 2014).

The Open Academic Analytics Initiative is aimed at researching the issues related to early warning systems that are applicable to most higher education institutions in the US (Jayaprakash et al., 2014). Their research has addressed issues related to the accuracy of timely predictions and the generalizability of predictive models. Jayaprakash et al. (2014) suggested that future research should examine the generalizability of predictive models across courses with different delivery formats (e.g., courses offered for on-campus students and courses offered for online students). This study extended Jayaprakash et al.'s (2014) work by using ensemble techniques to predict at-risk students at four points of time during a course period.

Based on these drivers, the investigations of issues related to predictive models intended to improve students' retention seem significant. The contribution of this study was to demonstrate the suitability and value of ML techniques when applied to predict students who are at-risk. In addition, this study added to the evaluation of the accuracy of timely prediction and the generalizability of predictive models.

Chapter 2

Review of the Literature

Machine Learning Techniques

Machine Learning is a subfield of artificial intelligence that focuses on the development of computational techniques and algorithms that allow computers to learn from prior experience without the need for any programming efforts (Samuel, 1959). ML techniques can be categorized into three main types of learning: supervised learning, unsupervised learning, and reinforcement learning (Russell & Norvig, 2009).

- Supervised Learning: In supervised learning, the learning system observes a set of labeled training examples that consist of feature-label pairs, $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, and learns a function that relates output to input. The function is then used to predict the label y for new input with features x (James, Witten, Hastie, & Tibshirani, 2013; Russell & Norvig, 2009).
- Unsupervised Learning: In unsupervised learning, the learning system observes a set of unlabeled training examples that consist of features only, $\{x_1, x_2, \ldots, x_n\}$, without any associated label y to identify patterns and clusters in the observed examples. The most frequent task in unsupervised learning is clustering (James et al., 2013; Russell & Norvig, 2009).
- Reinforcement Learning: In reinforcement learning, the learning system interacts with a dynamic environment to perform an action and receive a reward while it learns to maximize its future rewards (Russell & Norvig, 2009).

Since the focus of this work is on classification techniques, which is a supervised learning method, the following sections present a review on the supervised ML algorithms that were used in this research.

Logistic Regression

Logistic Regression (LR) is a classification method that is used to describe the correlation between a discrete response variable and one or more independent variables. While linear regression is usually used to predict response variables with continuous values, LR is used for classification problems where the response variable has two or more classes. LR has become one of the most common methods for classification problems in various domains (Hosmer, Lemeshow, & Sturdivant, 2013; James et al., 2013; Russell & Norvig, 2009). LR can be used to predict a binary response variable using multiple predictors that can have numeric or discrete values. LR uses the logistic function (Equation 2.1) to transform its output to a probability value between zero and one, which can be mapped to a class membership depending on a defined threshold.

$$p(X) = \frac{e^{\beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p}} \quad (2.1)$$

Equation 2.1 above shows the logistic function that is used to represent the probability of the response variable, where $X = (x_1, \ldots, x_p)$ are p features, and $\beta_0, \beta_1, \ldots, \beta_p$ are the regression coefficients. The LR uses the maximum likelihood method to estimate the regression coefficients (Hosmer et al., 2013; James et al., 2013). One of the main advantages of the LR method is that it allows the interpretation of the final model since the regression coefficients illustrate how the inputs affect the response variable. However, the performance of LR techniques decline when a non-linear relationship exists

between the variables and when the dataset contains missing values (Hosmer et al., 2013).

To demonstrate the application of LR to at-risk classification problems, let us consider that we are given a training data set on student final grades, and our goal is to predict whether a student will be at-risk or not based on the student's GPA and attendance. We have two inputs (GPA and attendance) and a response variable with two classes. "At-risk" class will be coded as 1 and "good standing" class will be coded as 0. To predict whether a particular student (test data) will be at-risk, LR fits a logistic model that maps the inputs to a probability of at-risk and returns a probability score between zero and one. To convert the probability value to a binary class, we define a threshold value where a probability value above the threshold will be classified as class "1," and a probability value below the threshold will be classified as class "0." For example, if our threshold was set to 0.6, a student whose probability of at-risk is 0.75 will be classified as "at-risk;" whereas, a student whose probability of at-risk is 0.55 will be classified as "good standing."

Decision Trees

Decision Tree (DT) is considered one of the most popular and simplest ML algorithms that has been successfully applied to a wide range of prediction tasks. DTs can be applied to both classification and regression problems. In DTs, the classifier is represented as a tree structure that models the relationship between the attributes and the potential outputs. Classification trees can also be represented as a set of if-then rules (Mitchell, 1997; Russell & Norvig, 2009). DTs are built using a recursive partitioning method (also known as divide and conquer) that splits the example set into subsets

starting at the top of the tree (root node) and ending at the bottom of the tree (leaf node). The root node is the first partition of a classification tree from which outgoing branches connect multiple nodes. Nodes with incoming and outgoing branches are called internal or test nodes, while nodes that only have incoming branches are known as leaves, which represent a final classification decision. Branches represent the conjunctions of features that lead to the leaf nodes. At each step of the DT construction process, a single feature that maximizes the value of some splitting criterion and effectively splits the example set into subsets is considered to create a decision node. The DT algorithm continues the recursive splitting on the smaller subsets, selecting the best candidate attribute to create another decision node, until all the training examples have been classified or a stopping criterion is reached (James et al., 2013; Russell & Norvig, 2009; Quinlan, 1993).

There are many advantages to using DTs for classification problems. First, DTs are highly interpretable since they can be displayed graphically as trees, and a set of if-then rules that are easy to understand can be generated from the resulting trees. In addition, DTs can easily handle different types of variables including categorical and continuous variables. Furthermore, unlike logistic regression, DTs can effectively handle missing data. However, DTs can suffer from model instability since a small change in the data can result in a large change in the structure of the tree (James et al., 2013; Kuhn & Johnson, 2013).

There are many implementations of DTs methods such as Quinlan's Iterative Dichotomiser (ID3) algorithm (Quinlan, 1986), C4.5 (Quinlan, 1993), C5.0, and Classification and Regression Trees (CART) (Breiman, Friedman, Stone, & Olshen 1984). One of the most widely used algorithms for classification trees is Quinlan's C4.5

algorithm (Quinlan, 1993), which is an improvement to his ID3 algorithm. For this study, Quinlan's C5.0 algorithm was used to represent the DTs classifier. This algorithm was developed by Quinlan as an improved version of the C4.5 algorithm, which uses the same DT induction technique but offers some additional features such as boosting and variable misclassification costs, in addition to some improvements in accuracy, memory usage, and processing speed (RuleQuest Research, 2017). The C5.0 algorithm uses gain ratio as the default splitting criterion during the tree construction process. Gain ratio is a measure based on information theory that is used at each test node to determine the feature that offers the best splitting outcomes. To explain how gain ratio is calculated, we start by finding the value of a purity measure called information (also known as entropy) and measured in bits. Let *D* denote a given training data set, the information for *D* can be expressed as:

$$Info(D) = -\sum_{j=1}^{C} p(j) \times log_2\, p(j) \tag{2.2}$$

where *C* represents the number of classes in *D* and *p(j)* refers to the proportion of cases in *D* that belong to the *j*th class. When applied to *D*, *info*(*D*) measures the average amount of information needed to identify the class of an example in *D*. The information gain for a feature *T* with *k* distinct values can be expresses as:

$$Gain(T) = Info(D) - \sum_{i=1}^{k} \frac{D_i}{D} \times Info(D_i)\,. \tag{2.3}$$

The information gained by *T* is significantly influenced by the number of outcomes and is maximal when there is one instance in each subset D_i. Information gain was the default

splitting criterion for the ID3 algorithm. However, the information gain measure is biased towards attributes with large numbers of possible outcomes such as attributes that contain identification numbers. Consequently, gain ratio is used as the default splitting criterion for the C4.5 and C5.0 algorithms since it overcomes the bias of information gain by using split information, which considers the number and size of subsets that result from splitting an attribute to normalize information gain. Split information can be expressed as:

$$Split(T) = -\sum_{i=1}^{k} \frac{|D_i|}{|D|} \times log_2 \frac{|D_i|}{|D|} . \quad (2.4)$$

The gain ratio can be found by calculating the ratio of a test's information gain to its split information and can be expressed as:

$$Gain\ ratio = \frac{Gain(T)}{Split(T)}. \quad (2.5)$$

Finally, the feature that achieves the maximum gain ratio is selected as the splitting node (Quinlan, 1993; Quinlan, 1996; Witten, Frank, Hall, & Pal, 2016).

Naïve Bayes

Naïve Bayes (NB) is a classification technique that is based on applying Bayes' theorem with the "naïve" assumption that the features are conditionally independent of each other given the value of the class variable. NB classifier is considered as simple, fast, and very effective compared to other sophisticated ML techniques. While NB is based on an assumption that is rarely true in real data sets, NB classification techniques have shown a competitive performance across a variety of classification problems.

Furthermore, NB can perform well with missing data (Russell & Norvig, 2009; Witten et al., 2016; Zhang, 2004).

As stated earlier, NB is based on Bayes Rule. Given a set of training examples with class labels, in which each example is represented by an input vector $X = (x_1, \ldots, x_p)$ with p predictors and a classification variable Y with two classes, i.e. for the at-risk context, we can label the data with "yes" being at-risk and "no" being in good standing. According to the Bayes' theorem, the probability of X being class k is

$$Pr(Y_k|X) = \frac{Pr(X|Y_k)\ Pr(Y_k)}{Pr(X)}, \tag{2.6}$$

where:

- $Pr(Y_k|X)$ is referred to as the posterior probability of the target class Y_k given the predictors X.
- $Pr(X|Y_k)$ is referred to as the likelihood, which is the conditional probability of predictors X given class Y_k.
- $Pr(Y_k)$ is the prior probability of the class outcome.
- $Pr(X)$ is the probability of the predictor values.

An example will be classified as at-risk ($Y=yes$) if and only if

$$Pr(Y_{yes}|X) > Pr(Y_{no}|X). \tag{2.7}$$

Therefore, the predicted class of the examples will be based on the class k that maximizes $Pr(Y_k|X)$. We assign an example with features vector X to the class Y_k for which $Pr(Y_k|X)$ is highest. The class Y_k for which $Pr(Y_k|X)$ is maximized is referred to as the maximum

posteriori (MAP) hypothesis (Russell & Norvig, 2009). As a result, equation 2.6 takes the form:

$$f(x) = argmax\ Pr(Y_k|X) = argmax\ \frac{Pr(X|Y_k)\ Pr(Y_k)}{Pr(X)}. \quad (2.8)$$

The denominator, *Pr(X)*, is a normalizing term that is used to return a probability value between 0 and 1 and is constant for all classes. Since the numerator terms ($Pr(Y_k|X)$ $Pr(Y_k)$) are the only ones we need to find the maximum posteriori hypothesis, we can drop the denominator and equation 2.8 becomes:

$$f(x) = argmax\ Pr(Y_k|X) = argmax\ Pr(X|Y_k)\ Pr(Y_k). \quad (2.9)$$

Finally, to facilitate the computational difficulties that are introduced with a larger number of features, the assumption of NB, which states that the features are conditionally independent of each other given the value of the class variable, is applied, and the final NB equation can be defined as:

$$f(x) = argmax\ Pr(Y_k|X) = argmax\ Pr(Y_k) \prod_{i=1}^{p} Pr(x_i|Y_k), \quad (2.10)$$

where *p* is the number of features, and *k* is the target class. The predicted class of a new example is the prior probability $Pr(Y_k)$ for a target class multiplied by the product of the independent likelihoods (Kuhn & Johnson, 2013).

Artificial Neural Networks

Artificial Neural Network (ANN) algorithms are computational models inspired by the biological neural networks that model the relationships between a set of inputs and outputs. An ANN is composed of connected nodes or units called artificial neurons in

which each connection has an associated numeric weight. The weight indicates the strength of the connection and changes as learning proceeds. Each connection between the nodes can transmit a signal that can be received by a connected node to be processed and transmitted again to connected nodes. Typically, an ANN model is comprised of three different layers: the input, hidden, and output layers. The input layer is composed of nodes in which each node represents a single feature of the input feature vector. The hidden layer processes the data received from the input nodes and transmits it to the output layer. The output layer takes the information received from the hidden layer and computes the predicted value of the output. In feed-forward networks, each node receives input only from nodes in the nearest preceding layer (Russell & Norvig, 2009). Figure 2 below depicts a simple three-layer, feed-forward neural network.

In general, a connection from node *i* to node *j* transmits the activation a_i (signal) from *i* to *j*. Also, as mentioned earlier, each connection from node *i* to node *j* has a numerical weight $w_{i\,j}$ that indicates the strength of the connection. Additionally, each node *j* has a dummy input x_0, which is always equal to 1 with associated weight $w_{0,j}$. Then, each node *j* uses an activation function to compute an output signal that can be sent to each connected node in the succeeding layer. To produce the activation function, each node *j* first computes a weighted sum of its inputs:

$$in_j = \sum_{i=0}^{n} w_{i,j} a_i. \tag{2.11}$$

Then, it uses an activation function *g* to the weighted sum to produce the output signal:

$$a_j = g(in_j) = g\left(\sum_{i=0}^{n} w_{i,j} a_i\right). \quad (2.12)$$

The sigmoid activation function is the most widely used function and results in an output between 0 and 1 (Russell & Norvig, 2009).

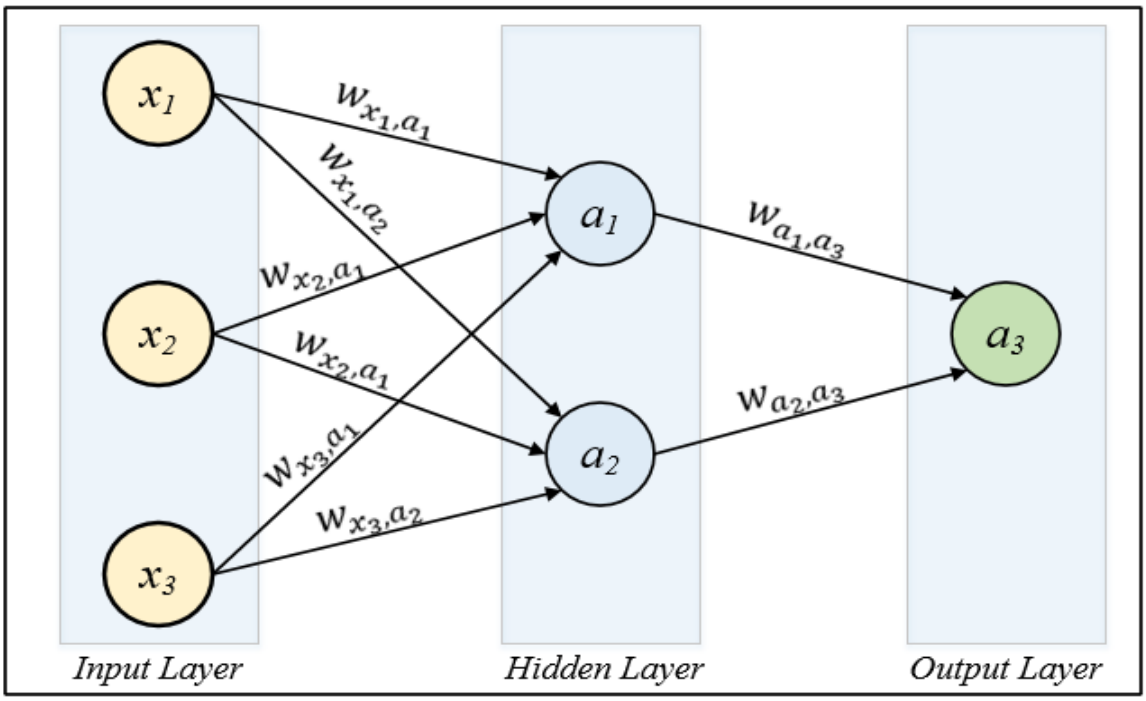

Figure 1. Feed-Forward Neural Network.

To explain how activation signals are derived, consider the simple network shown in Figure 1, which has three input nodes in which they represent an input victor x = (x_1, x_2, x_3), two hidden nodes, and one output node. The output signal at node a_1 is given by:

$$a_1 = g(w_{0,1} + w_{x_1,a_1} x_1 + w_{x_2,a_1} x_2 + w_{x_3,a_1} x_3) \quad (2.13)$$

Ensemble Classifiers

Ensemble classifiers combine the outputs of multiple classification algorithms to classify new observations. Researchers from diverse disciplines have explored and applied ensemble methods in a broad variety of domains. It has been found that ensemble

techniques constantly achieve better predictive performance than their base classifiers (Dietterich, 2000; Rokach, 2010). Fundamentally, the idea of ensemble methods is to train several diverse classifiers and use various techniques to combine their predictions in order to build an ensemble classifier that outperforms any single classifier. While there exists a considerable number of ensemble methods, bagging and boosting are the most widely used techniques in ensemble learning (Rokach, 2010; Russell & Norvig, 2009).

Bagging, short for bootstrap aggregation, is an ensemble technique that utilizes bootstrapping sampling method to train multiple classifiers (Breiman, 1996). Bagging generates *n* bootstrap samples by randomly selecting instances with replacement from the original training set. Then, *n* number of classifiers are trained using the *n* bootstrap samples with each bootstrap sample to train a single classifier. Finally, the final ensemble classifier is built using all the *n* classifiers by combining their predictions using a majority vote combining method (Rokach, 2010).

Boosting is another popular ensemble technique that combines many weak classifiers into a strong classifier. Adaptive boosting (AdaBoost) algorithm that was introduced by Freund and Schapire (1997) is the widely used implementation of boosting methods. AdaBoost algorithm is based on the idea of training weak classifiers that iteratively assign more weight to misclassified examples so that subsequent classifiers learn how to classify those difficult to classify examples. Initially, the AdaBoost algorithm starts by training a classifier using a training dataset in which all examples are equally weighted. Examples that were misclassified by the classifier get their weight increased for the next iteration, while the examples that were correctly classified get their weight reduced. This means that the correctly classified exampled will have less chances

to be selected in the training set for the subsequent classifier, while the misclassified examples will have higher chances of being selected in the following iteration. Additionally, a weight indicating the overall classification accuracy is assigned to each individual classifier. The algorithm performs multiple rounds of iterations in which a weak classifier is learned to classify a set of hard to classify examples. The process continues until a classifier is able to classify all examples, or the performance no longer improves. Finally, the algorithm generates a final classifier by combining all individual classifiers using their weights as a combination method. To classify a new observation, each classifier vote is weighted according to its classification accuracy (Rokach, 2010; Russell & Norvig, 2009).

Review of Research on Students' Performance Prediction

In the previous years, ML techniques have increasingly become one of the popular topics in education. There are an increasing number of ML applications in education, from predicting students' performance, modeling student behavior, improving assessment and feedback, recommending resources to students, and others. Of particular relevance to this study, this section reviews only research where the main focus was to predict students' performance as measured by final grades.

Early attempts to apply ML methods to predict students' academic outcomes and act upon it can probably be traced back to the early 2000s. One of the initial works detailing the use of genetic algorithms to optimize the prediction accuracy of the classifier was performed by Minaei-Bidgoli, Kashy, Kortmeyer, and Punch (2003). Minaei-Bidgoli et al. (2003) used features obtained from a logged data in a web-based

system to predict students' performance. The dataset included a total of 227 students' records who registered an introductory physics courses in the Spring semester of 2002. They treated the prediction problem as a classification task. Several experiments were performed to classify students into nine classes (i.e., 4, 3.5, ..., 0), three classes (i.e., high, medium, low), and two classes (i.e., passed, failed). Six different classifiers were compared in this study including quadratic Bayesian classifier, 1-nearest neighbor (1-NN), k-nearest neighbor (KNN), multilayer perceptron (MLP), and decision tree. Later, the prediction accuracy was evaluated and compared to determine the numbers of classes and algorithms that yield the best prediction accuracy. While their prediction accuracy varied between 51% and 83%, the results revealed that the KNN algorithm when applied to classify students into two classes outperformed the other approaches and achieved about 83% prediction accuracy. It was also noted that using the genetic algorithm to optimize the prediction accuracy of the classifiers improved the prediction accuracy by 10%. Finally, of the initial ten features that were included in the dataset, only six attributes were used by the algorithms as predictors of students' performance namely success rate, success at the first try, the time at which the student got the problem correct relative to the due date, the number of online interactions of the student both with other students and with the instructor, number of attempts before correct answer is derived, and total time spent on the problem.

This work was among the first to classify students by using genetic algorithms to predict their final grade. The prediction accuracy ranged from 62% for nine classes, 72% for three classes, to 94% for two classes. It must be noted that the two classes dataset suffered from a considerable imbalance (e.g., 72% passed, 28% failed), which requires

that the evaluation of algorithms to go beyond the traditional accuracy measure. However, prediction accuracy was the only metric used, which may suggest an imperfect conclusion.

In the same year, Kotsiantis, Pierrakeas, and Pintelas (2003) used several ML techniques to predict the performance of students who studied an online course, using demographic and learning performance attributes. The dataset included a total of 354 students' records and 11 attributes for students who were registered in an introductory course at the Hellenic Open University. Seven attributes represented the demographics attributes and were used for the first experiment. The other four attributes represented the students learning performance attributes. The second experiment included demographic attributes and the first available learning performance attribute. As data for the other three learning performance attributes became available, three more experiments were conducted constituting a total of five experiments that represent a specific time in the semester. These multiple experiments allowed the authors to compare the prediction accuracy among all five experiments to find out the earliest time with acceptable accuracy to predict at-risk students. Six ML techniques were compared namely, decision trees, neural networks, naïve Bayes, instance-based learning algorithms, logistic regression, and support vector machines. The comparison of the six algorithms showed that the naïve Bayes algorithm outperformed the other algorithms with an accuracy reaching 63% in the initial predictions based only on demographic data and exceeding 83% before the middle of the semester. It must be mentioned that the authors only used prediction accuracy as the only evaluation metric, and a more detailed discussion about the algorithms' performance should have been presented.

As an extension to their previous work from 2003, Kotsiantis, Pierrakeas & Pintelas (2004) compared the performance of the same ML techniques utilized in their previous work to predict students' performance using the same sets of attributes. Unlike their previous work, the goal of the recent work was to compare the algorithms performance in terms of multiple criteria including prediction accuracy, sensitivity, and specificity. They conducted two separate experiments. The first experiment used the records of all students registered in an introductory course (354) to train the algorithms, while the second experiment only used the records for a subset of students (28) to train the algorithms. The rationale for the second experiment was to test the performance of the algorithms using data that represents only a group of students who were registered in one section of that introductory course. They argued that the collection of students' records is time consuming and expensive. Once again, as it was in their previous study, naïve Bayes classifier was found to be the most accurate algorithm for students' performance predictions in both experiments. For the first experiment, the results showed that the performance of naïve Bayes algorithm was more than satisfactory with an overall accuracy of about 73%, an overall sensitivity of 78%, and an overall specificity of about 67%. As for the second experiment, the overall performance of naïve Bayes algorithm was less satisfactory than the accuracy in the first experiment with an overall accuracy of about 66%, an overall sensitivity of 73%, and an overall specificity of about 61%. The performance of the algorithms in the second experiment may be due to the limited number of instances.

In 2006, Al-Radaideh, Al-Shawakfa and Al-Najjar (2006) tested three different classification techniques (i.e. ID3, C4.5, and naïve Bayes) to predict the final scores of

students who studied the C++ course in Yarmouk University, Jordan. Students and lecturers background attributes were used to train the algorithms. Among the limited number of attributes, high school grade contributed the most to the classification of students in different groups. While only the prediction accuracy was used to measure the performance, all algorithms performed poorly with a maximum of 38% accuracy. Such poor performance might be due to the limited number of attributes and sample size.

Also in 2006, Calvo-Flores, Galindo, Jiménez, and Piñeiro (2006) used artificial neural networks to predict students' final grades based on LMS usage data. The dataset included LMS usage data of 240 students who were registered in one course at the University of Cordoba, Spain. The holdout method was utilized by the authors where 60% of the data was used to train the model, and 40% was held to test it. The results revealed that the artificial neural networks achieved an average of 80% accuracy rate. It could have been beneficial to provide more details on the analysis of the model performance and to use other criteria to evaluate the model performance with the accuracy rate. Also, it is worth noting that the prediction accuracy was based on data that represented the full semester, and it would be interesting to find if such accuracy can be achieved with data that represents only a portion of time in the semester, so at-risk students can be identified while there is still time to help them.

Simultaneously, Etchells, Nebot, Vellido, Lisboa, and Mugica (2006) used fuzzy inductive reasoning (FIR) and orthogonal search-based rule extraction (OSRE) techniques to construct a model to predict students' final grades. Unlike the previous works that were discussed earlier, this work focused on two main goals: to determine the most important attributes that can predict online students' final grades and to explain the

prediction task in the form of simple and interpretable rules. The dataset used in their experiments included data about 722 students who were registered in one graduate course. FIR was used to perform the feature selection process, which revealed that the average marks of the co-evaluation, the initial class plan, and the experience report attributes were the most relevant features to predict the final score for each student. Two experiments were conducted where the first one was based on all attributes in the datasets, and the second one was based only on the three attributes selected by FIR. In both experiments, the MLP algorithm was trained using only 50% of the data while the other 50% of the data was held for testing. Finally, OSRE was utilized in both experiments to extract rules from the trained MLP model. The results of both models were similar, which indicates that feature selection can help in achieving similar or higher accuracy with fewer rules for interpretation.

In 2008, Romero, Ventura, and Garcia (2008) compared the performance and usefulness of different data mining methods and techniques to classify students based on their LMS usage data and the final grades obtained in their respective courses. The dataset included records for a total of 438 students who were enrolled in seven different courses at Cordoba University, Spain. Different algorithms that represented statistical classifiers, decision trees, fuzzy rule learning, neural networks, and rule induction were tested and compared. While their dataset presented a clear imbalance, they addressed the problem of imbalanced data by using random over-sampling techniques that involve a random copying of the minority classes in the dataset until all classes have the same number of instances. The authors used stratified 10-fold cross-validation techniques to evaluate each algorithm. In addition to the classification accuracy, geometric mean of

accuracies per class was used to evaluate the performance of each algorithm. The results revealed that the decision tree algorithms achieved the highest classification accuracy rate among the other algorithms. However, the decision tree algorithm was only able to achieve 67% accuracy rate. Such unsatisfactory accuracy rate was due to the fact that the datasets included missing values because of the differences in activities among the seven courses. Moreover, it was found that the performance of the algorithms when using the original dataset was better than when over-sampling techniques were used.

In 2009, Lykourentzou, Giannoukos, Mpardis, Nikolopoulos, and Loumos (2009) used neural networks and multiple linear regression techniques to classify students into two groups based on their final grades. The grades of four multiple-choices tests were used as inputs to train the algorithms. The dataset included the results of 57 students who studied an introductory online course in the Spring of 2006 and 2007. The number of students who studied the course in the Spring of 2006 was 32 while 25 students studied the course in the Spring of 2007. A total of three experiments were conducted where the first experiments only included the first and second multiple-choice test grades. The second experiment included the results of the first three multiple-choice tests, while the third experiment included all attributes. The three experiments were intended to test the applicability of ML techniques to address the need for early and dynamic prediction of students' performance. To train the algorithms, 85% of the Spring 2006 dataset was used while the remaining 15% was held to validate the models. The Spring 2007 dataset was used to test the models. The decision to use the Spring 2006 dataset for the training stage while using the spring 2007 dataset for the test stage was taken to assess whether the algorithms were able to predict the performance of future students accurately. The

algorithms performance was evaluated and compared in terms of correlation coefficient and mean absolute error. The results revealed that neural network models outperformed linear regression models in all prediction stages. While the authors concluded that the prediction accuracy improved as the course progressed, it was found that predictions with a satisfactory accuracy can be made as early as the third week of a 10-week program.

Additionally, Thai-Nghe, Busche, and Schmidt-Thieme (2009) compared some ML techniques that were used to predict students' final performance. Their primary goal was to improve the prediction accuracy when datasets with class imbalance were used to train the models. Two datasets were used to classify students into Pass/Fail groups. The first datasets contained a total of 20,492 instances and 14 attributes, while the second dataset only consisted of 936 instances and 14 attributes as well. Three techniques have been applied to deal with the class imbalance problem. The first method utilized an over-sampling method called the synthetic minority over-sampling technique (SMOTE). The second approach utilized cost-sensitive learning (CSL) to build the model with minimum misclassification costs. The third method combined SMOTE and CSL. It must be noted that this work was among the first to address the problem of misclassification costs. The costs of misclassifying the actual "fail" students to "pass" students are much costlier than the false alarm since at-risk students would be expelled from the university if they keep failing their courses without getting enough help. The authors considered the use of multiple classifiers such as decision trees, Bayesian networks, and support vector machines. Also, they used AUC, F-measure, and total cost to evaluate the classifiers on the proposed three methods. The results of their work revealed that all three methods presented satisfactory results. The decision tree models were found to perform better than

the other techniques when the larger datasets were used, while the support vector machine models outperformed the other algorithms when the smaller datasets were used. Furthermore, unlike Romero et al.'s (2008) work, the authors concluded that the use of different techniques to address the class imbalance problem resulted in an improved prediction accuracy.

In the same year, Zafra and Ventura (2009) proposed the use of a grammar guided genetic programming algorithm (G3P-MI) to predict if students would fail or pass a certain course based on their LMS usage data. The proposed algorithm was compared with other multiple instance learning techniques, such as algorithms based on rules, decision trees, support vector machines, naïve Bayes, logistic regression, and neural networks. The dataset included records for a total of 419 students who were enrolled in seven different courses at Cordoba University, Spain. The attributes used in this study were related to quizzes, assignments, and forums. The performances of the algorithms were compared in terms of prediction accuracy, sensitivity, and specificity. The results of this study showed that the proposed algorithm (G3P-MI) outperformed the other algorithms and achieved a satisfactory prediction accuracy of about 75% while it maintained a good sensitivity and specificity balance (70%, 77%), which was considered a good tradeoff. The authors also argued that their proposed algorithm provided a comprehensible model that would enable the correlation of certain LMS activities and the time devoted to them with the final grades obtained in the course. Lastly, since this study is based on data that was collected at the end of semester and comprised all activities performed during the entire semester, the authors pointed to this limitation and

emphasized the need to investigate the possibility of predicting students' performance early in the semester so that enough help can be provided to at-risk students.

In 2011, Zafra, Romero, and Ventura (2011) extended the work of Zafra and Ventura (2009) and compared the performance of traditional supervised learning algorithms and multiple-instance learning algorithms when applied to students' performance prediction problems. While the earlier work compared the performance of MIL algorithms with a proposed algorithm based on MIL, the more recent work focused on emphasizing the differences between single-instance learning and multiple-instance learning. The same dataset that has been used in the previous work (Zafra & Ventura, 2009) has been used again in this work to train the algorithms. The datasets included information stored about three activities in LMS, including quizzes, assignments, and forums, for a total of 419 students enrolled in seven different courses. The dataset included diverse information about students and courses where each student was represented by a variable number of instances depending on his/her work, and each course was represented by activities that were available for that course. In this dataset, a hardworking student would have a higher number of instances, while a lazy student may have a lower number of instances. Similarly, a course may be represented by all three activities as attributes; other courses may be represented by only one activity. In single-instance learning, such missing activities will be treated as missing values, while in multiple-instance learning, each student will be represented by the information available about that student, which eliminates the missing values problem. Algorithms based on rules, decision trees, support vector machines, naïve Bayes, logistic regression, and neural networks that represented both traditional supervised learning and multiple-

instance learning were applied and compared. The results of their experiment showed that the performance of algorithms based on multiple-instance learning was significantly better than the performance of algorithms based on a traditional supervised learning. Moreover, the performance of algorithms based on multiple-instance learning in terms of accuracy, sensitivity, and specificity ranged from 66%-74%, 71% - 86%, to 43% - 64%, respectively, while the performance of algorithms based on a traditional supervised learning in terms of accuracy, sensitivity, and specificity ranged from 58%-70%, 70% - 89%, to 36% - 64%, respectively.

In 2012, Barber and Sharkey (2012) reported on the development of predictive models for the University of Phoenix to identify academically at-risk students. The models combined data from the LMS, financial aid system, and student information system to classify students into two groups (i.e., high risk, low risk). The data included basic demographic information, academic history within the University (such as number of transfer credits, number of courses taken, and percentage of points earned in these courses), and LMS data (including forum postings, points earned by week within the course, and assignment submissions). Two models were developed and validated. The first model was developed using logistic regression where 50% of the data was used to build the model, while the remaining 50% was used as hold-out to validate the developed model. Separate models were developed for each degree level. The authors reported on the performance of the models from course week zero to course week four. Week zero's model only used previous information, such as demographic and academic history, to classify students, while models one to four added LMS usage data as they became available the following week. The results revealed that in week zero, the model was able

to achieve 50% accuracy, while in the following weeks, it achieved an average accuracy of 94% with no week below 85%. The second model was developed using a naïve Bayes algorithm and was validated using 10-fold cross validation techniques. The results showed that the naïve Bayes model was able to accurately classify 85% of all students at week zero, while it achieved 95% classification accuracy by week three. While prediction accuracy was the only measure to evaluate the models, it is worth noting that this work is considered among the few that addressed the need for an early and dynamic prediction of students' performance. Furthermore, despite the fact that most of the previous works were based on data for a limited number of students and courses, this work indicated that there are differences in student populations, courses, and degrees, which suggest that special predictions models should be developed accordingly to account for these differences.

Also in 2012, Kovacic (2012) investigated the possibility of classifying students into two group (pass, fail) using a dataset that included demographic attributes (e.g., age, gender, ethnicity, education, work status, and disability) and study environment attributes (e.g., students' program of study and course semester). Moreover, the dataset included data from about 450 students who were registered in a one semester information systems course during 2006 to 2009. Despite the limited number of attributes, feature selection techniques were used to rank the attributes according to their ability to predict students' final performance and showed that the most important factors that separate successful from unsuccessful students were ethnicity, program of study, and course semester. Models based on decision tree and logistic regression were developed. The results revealed that models based on decision tree outperformed the logistic regression models

and achieved about 61% classification accuracy. Such unsatisfactory results are similar to the results obtained by Al-Radaideh et al. (2006), which may suggest that using demographic data by itself or with a limited number of attributes is not enough to separate successful from unsuccessful students.

Additionally, Smith, Lange, and Huston (2012) reported on the use of data-mining techniques to predict students' performance in order to establish an early-warning system for at-risk students at Rio Salado Community College. The dataset included the records of 539 students who were enrolled in one online freshman accounting course. Also, students' records included demographic data, past academic history, and LMS usage data (such as logging in to the LMS, opening a lesson, completing an assessment, and viewing a grade). This study also weighed recent activity more heavily than activities from earlier weeks of the course. The naïve Bayes classification method was the only technique used to predict at-risk students. The authors concluded that a strong correlation existed between LMS activity and students' final performance. Furthermore, the inclusion of weighting recent activities resulted in an increased accuracy in predicting students' final performance as information accumulates over the duration of the course.

The most comprehensive study in terms of tested classifiers, feature selection algorithms, and performance measures was performed by Romero, López, Luna, and Ventura (2013). They implemented 14 classification algorithms, including several variants of rule-based algorithms, tree-based algorithms, function-based algorithms, and Bayes-based algorithms. Moreover, this work also used a wide range of feature selection algorithms, such as information gain, correlation-based feature selection methods, OneR, RELIEF, and support vector machines. Furthermore, the classification algorithms

performances were evaluated in terms of accuracy and F-measure (harmonic mean of precision and recall). The algorithms were run on a relatively small dataset that was obtained from LMS usage data and only represented participation in online discussion forums. The dataset included the records of 114 undergraduate students during a first-year course in computer science in 2011–2012. Each student record included a total of nine forum participation indicators, which were categorized into three types of variables based on forum activity: quantitative, qualitative, and social. This study focused on three main goals: (i) investigating the possibility of making an early prediction of students' academic outcome, (ii) identifying the most important attributes that best predict students' performance, and (iii) examining the effect of the messages' quality on prediction accuracy. In order to fulfill these objectives, the authors conducted three tasks that involved: (i) instances selection based on participation time (in the middle and at the end of the course), (ii) instances selection based on the quality of the messages (only messages related to the course), and (iii) attribute selection as a result of the feature selections algorithms. These three tasks resulted in eight different datasets as shown in table 1.

Table 1. Datasets Used by Romero et al. (2013)

Dataset	Collection Time	Instances Used	Attributes Used
Dataset 1a	Middle of the course	All	All
Dataset 1b	Middle of the course	All	Only selected by feature selection algorithms
Dataset 2a	Middle of the course	Only with messages related to the course	All

Dataset 2b	Middle of the course	Only with messages related to the course	Only selected by feature selection algorithms
Dataset 3a	End of the course	All	All
Dataset 3b	End of the course	All	Only selected by feature selection algorithms
Dataset 4a	End of the course	Only with messages related to the course	All
Dataset 4b	End of the course	Only with messages related to the course	Only selected by feature selection algorithms

A 10-fold cross-validation technique was used across all classifiers for evaluation. While the results of all algorithms were presented in the study, only those for the algorithms with the best performance were discussed in this work, namely, SMO and naïve Bayes. The results showed that SMO and naïve Bayes were the best classification algorithms that obtain the highest accuracy and F-measure values in most datasets. SMO obtained a better accuracy and F-measure values in 3 out of 8 datasets, whereas naïve Bayes obtained better accuracy and F-measure in 4 out of 8 datasets. In addition, the results indicated that two quantitative attributes (the number of messages sent and the number of words written), together with the only qualitative attribute (the average evaluation obtained in messages) and the two social network variables (the degree of centrality and the degree of prestige) were the most important features for predicting the students' final performance based on their usage data from discussion forums. Furthermore, algorithms which used this set of selected attributes instead of the full set of attributes achieved the highest accuracy and F-measure values. Also, algorithms that used the dataset that included only messages with content related to the course subject

improved the accuracy of all the algorithms in all cases. Finally, while the data collected at the end of the semester provided higher accuracy, it was shown that the accuracy obtained by using data collected at the middle of the course was satisfactory to be used as an early warning system.

In 2014, Hu, Lo, and Shih (2014) studied the application of ML to develop an early warning system that could predict student performance using time-dependent variables obtained from an LMS. The dataset included the records of 300 undergraduate students who studied the information literacy and information ethics online courses in a national university in Taiwan. Each student record included a total of 14 attributes, which were categorized into four types of variables: login behavior, the use of online course materials, assignment status, and discussion status in the forum. To understand the effects of time-dependent variables on students' performance, data about students' online behaviors was collected at three different times during the course (weeks four, eight, and thirteen) to develop prediction models using algorithms based on decision tree and logistic regression. Also, they used ensemble classifiers to enhance the predictive power of the previous classification techniques. The performance of the classification models was evaluated and compared using three metrics, including prediction accuracy, type I error, and type II error. The results showed that models based on decision tree outperformed models based on logistic regression and provided better accuracy in classifying students based on their online behaviors. Moreover, it was revealed that ensemble techniques, when applied to classification tree improved the prediction accuracy. Lastly, this study supports the claim that regardless of the classification

techniques used, the accuracy of an early warning system is improvable by considering time-dependent variables.

In that same year, Jayaprakash, Moody, Lauría, Regan and Baron (2014) reported on the Open Academic Analytics Initiative which aimed at researching the issues related to early warning systems that are applicable to most higher education institutions in the US. Their research has addressed issues related to the accuracy of timely predictions and the generalizability of predictive models. Jayaprakash et al. (2014) predicted at-risk students at three points in time during a course. While the prediction accuracy ranged from 65% to 85%, they found that the prediction accuracy increased as the course progressed. Jayaprakash et al. (2014) also addressed the issue of the generalizability of predictive models. They trained a predictive model on data from Marist College and applied the predictive model across four other institutions. Although the prediction accuracy of the predictive model was 10% lower when applied to the other four institutions, the researchers claimed that the generalizability of the predictive model was higher than expected. Jayaprakash et al. (2014) suggested that future research should examine the generalizability of predictive models across courses with different delivery format (e.g., courses offered for on-campus students and courses offered for online students).

Chapter 3

Methodology

Introduction

This study used ML techniques to predict students who may be at-risk based on available student data. The predictions needed to be early in the semester so appropriate measures can be applied to help at-risk students succeed in the course. The prediction task was treated as a binary classification task to determine whether students are in good standing or at-risk. The target attribute that was employed to classify students in good standing and at-risk was based on students' grades in a particular course, where students with a grade worse than "C" was considered at-risk. The R platform (R Core Team, 2017) was used to perform this study. R is a free software environment for mathematical and statistical computations and graphics that provides a variety of implementations for machine learning algorithms and numerous data mining tasks. In this study, the Classification and Regression Training (*caret*) package (Kuhn, 2017) was extensively utilized for the execution of all steps in the research process. The *caret* package provides a unified interface that streamlines the process for training and evaluating predictive models since it contains tools for the preparation, training, evaluation, and visualization of ML models and datasets. Figure 2 depicts an overview of the study approach. In the following sections, a description of each step of the research process is presented.

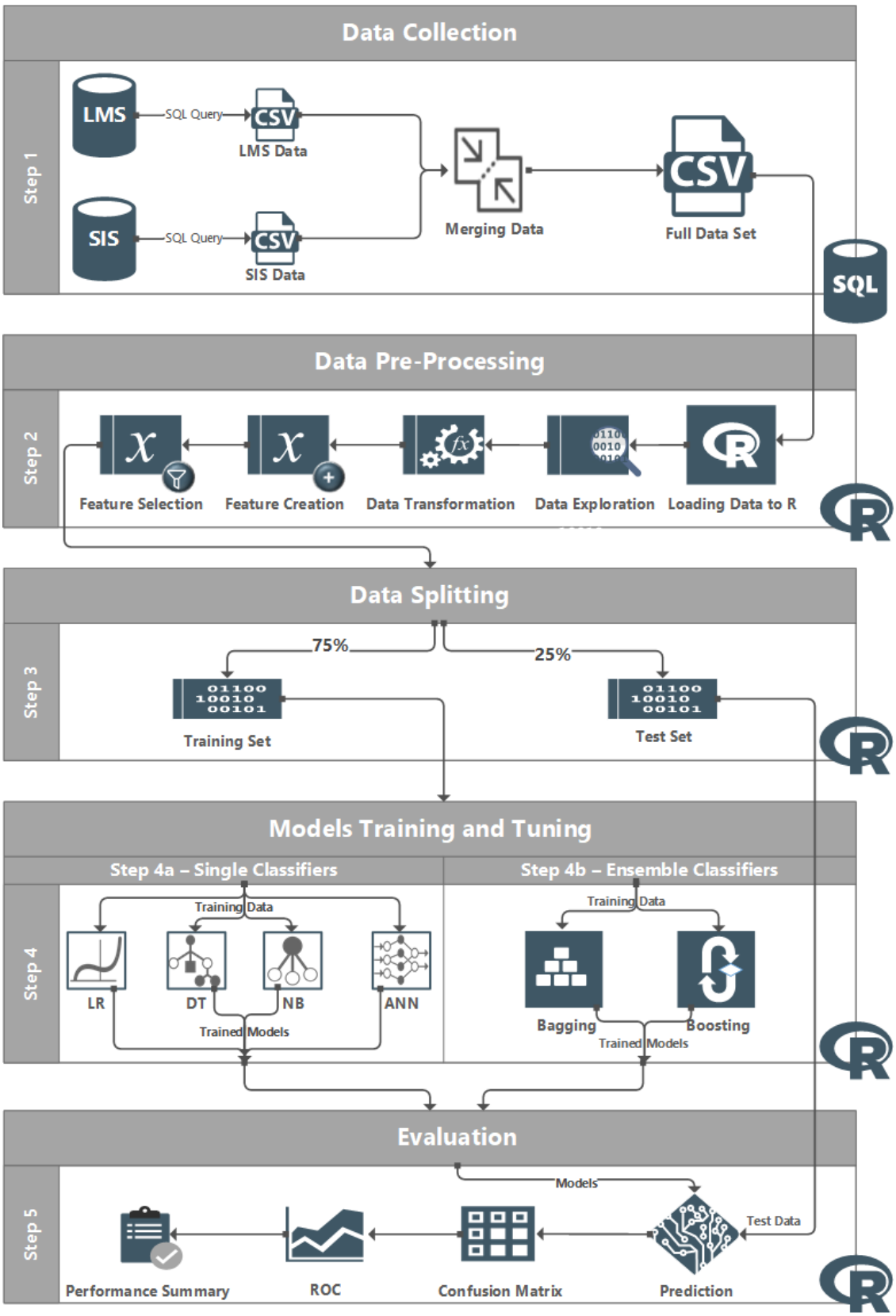

Figure 2. Overview of the Research Methodology.

Step 1: Data Collection

The data was collected from Jazan University, Saudi Arabia. Given my affiliation with Jazan University, I had access to the necessary data for this study. Approval for this study was granted by the University President. Additionally, two contact persons were appointed to help in obtaining the data. Students' data was obtained from the student information system (SIS) and the learning management system (LMS). SQL queries were used by the contact persons to obtain the data from the databases and was provided to the researcher as comma-delimited text (CSV) files. Data for this study included a sample of on-campus and online students enrolled during Fall 2016, Spring 2017, and Fall 2017 semesters in courses that used the learning management system. Data was collected at different stages of the courses during a 16-week semester (e.g., 4th week, 8th week, and 12th week) so the prediction accuracy of timely predictions could be assessed and compared. The LMS contact person provided the researcher with data files that included LMS log data of students enrolled in courses that utilized the LMS. The researcher then prepared a list of courses that utilized the LMS during the three semesters and sent it to the SIS contact person. Based on the courses list, the SIS contact person provided the researcher with data files that included demographic and academic data about the students and courses.

Students' data included demographic and aptitude data (e.g., gender, age, and standardized high school test scores), academic and course related data (e.g., GPA, course level, and attendance), LMS interaction data (e.g., LMS site visits, resources views, forum participations, and assignment submissions), and LMS grade book data

(e.g., assignments and tests grades). Table 2 and table 3 provide a detailed list of features that were obtained from the SIS and LMS.

Table 2. Features that were obtained from the SIS

Type		Variable
Student Demographic Data	1	Birth Year
	2	Gender
	3	Nationality
	4	International Student Indicator
Student Admission Data	5	High School GPA
	6	High School Graduation Year
	7	High School District
	8	General Aptitude Test Verbal Score
	9	General Aptitude Test Quantitative Score
	10	General Aptitude Test Total Score
	11	Scholastic Achievement Admission Test Score
	12	Scholarship Type
	13	Starting Term
	14	Degree
	15	Study Type
	16	College
	17	Major
	18	Transferred Student Indicator
Student Academic Data	19	Cumulative GPA
	20	Last Semester GPA
	21	Total Registered Hours
	22	Total Earned Hours
	23	Total Transferred Hours
	24	Last Semester Registered Hours
	25	Last Semester Earned Hours
	26	Current Level
	27	Academic Standing
	28	Current Semester Registered Hours
	29	Current Semester Registered Courses
Course Information	30	Course College
	31	Course Department
	32	Course Level
	33	Course Hours
	34	Course Schedule Time
	35	Course Total Registered Students
	36	Course Classroom
	37	Course Instructor
Student-Course Information	38	Final Grade (Target Variable)

Table 3. Features that were obtained from the LMS

Type	Variable	Description	Availability
Systems Usage	Sessions	The total number of times the student logged into the course.	Sessions per week
Learning Tools & Resources Usage i.e.: - Assignments - Virtual Lectures - Media Lectures - Quizzes - Forums	Learning tool – opened	The total number of learning tool opened or viewed by a student. If a student opens the same learning tool multiple times, the system records each entry.	Counts per week
	Learning tool – time spent	The total time a student spends within the learning tool from initiation of the tool until the student leaves the assessment.	Total minutes per week
	Learning tool – completed	The number of learning tools completed by the student.	Counts per week
	Forum postings created	The total number of discussion postings created by the student within the course.	Counts per week
	Learning resource – downloads	The total number of downloads for each resource per student.	Counts per week
Students Performance Data	Attendance	The total number of attended session divided by the total number of class sessions.	Attendance percentage per week
	Grade Book Data	The total points a student earn or lose in all assessments including assignments, quizzes, and exams.	Points earned per week Points lost per week

In table 2, the SIS features are not time dependent and could be available as soon as the course starts. Therefore, SIS data was obtained once and used repeatedly with all experiments. However, as in table 3, the features obtained from the LMS were time dependent and were available in weekly totals. Since the data was collected at different stages of the course to investigate the viability of early identification of at-risk students, the predictions were based on data available 4, 8, and 12 weeks into the course.

Step 2: Data Preprocessing

The data was not provided to the researcher in a format ready for training the predictive models. Therefore, a data preparation task was undertaken to prepare the data for ML algorithm applications. Data preprocessing is an essential step to produce datasets that can be used by the classification techniques. It is important to note that the prediction accuracy of the ML algorithms depends on the quality and reliability of the available data. The preprocessing phase involved data cleaning, data transformation, variables creation, and features evaluation and selection.

Data Cleaning and Transformation

In this phase, a thorough examination of attributes and their corresponding values was performed to reduce any irregularities in the data, handle missing values in the data, and enhance the reliability of the data. Additionally, the preprocessing phase involved transforming the input data into a form that is preferred by the ML algorithms. Real data contains input features with values that vary in their numeric ranges. As a result, attributes with greater numeric ranges could have a larger influence on the learning algorithm than the attributes with less numeric ranges, which may impact the

classification accuracy of the algorithm. Some algorithms, such as artificial neural network, work best when the features values are scaled to a small range near zero (Kuhn & Johnson, 2013). One of the most common methods for transforming input data is z-score standardization which scales different features' values such that they follow a standard normal distribution. Consequently, the transformed features will have a comparable range or scale of measurement, such that none will have more influence than the others on the learning algorithm. As shown in equation 3.1, standardization of an element in feature X involves subtracting the mean value of feature X, and then dividing the outcome by the standard deviation of feature X.

$$X_{transformed} = \frac{X-\mu}{\sigma} = \frac{X-Mean(X)}{SD(X)}. \quad (3.1)$$

The *preProcess* function in the *caret* package provided a simple interface to perform several preprocessing tasks, including features transformation. As shown in the code below, the centering and scaling method was used to standardize the input features.

```
> library(caret)
> StudentsData <- read.csv("StudentsData.csv")
> preProcValues <- preProcess(StudentsData, method = c("center", "scale"))
> StudentsDataTransformed <- predict(preProcValues, StudentsData)
```

Feature Creation

In the variable creation process, some of the variables were used to derive new variables (e.g., summed attributes for weeks 5-8, summed attributes for weeks 9-12, week 8 improvement index, and week 12 improvement index). The summed attributes for weeks 5-8 were derived by subtracting the value of the summed attributes for weeks 1-4 from the value of the summed attributes for weeks 1-8. Similarly, the summed attributes

for weeks 9-12 were derived by following the same process. The summed attributes for weeks 5-8 and weeks 9-12 were created to have a better representation of the students' overall performance as compared to the previous stage. Additionally, week 8 improvement index for a specific attribute was derived by subtracting the value of the summed attributes for weeks 4-8 from the value of the summed attributes for weeks 1-4. Similarly, week 12 improvement index for a specific attribute was derived by following the same process. Intuitively, it was expected that a positive and greater value of the improvement indexes would have a positive impact on the students' final performance. Multiple experiments were conducted to examine the obtained variables and create new aggregated and derived variables. An evaluation of the derived variables was performed and only those variables that lead to better prediction accuracy were kept in the final variable set.

Feature Selection

Feature selection is the process of selecting a suitable subset of relevant and informative features that can be used to construct a model that can achieve an equal or better accuracy than models constructed with the full features set. While the performance of ML algorithms is significantly dependent on the quality of the selected features, eliminating redundant or irrelevant features results in a reduction in the training time and computational cost of the ML algorithms, a reduction in the model complexity, in addition to the improvement in the model performance. The feature selection methods can be classified into two main categories: filter and wrapper methods. The filter feature selection methods rely on the general characteristics of the data itself and use statistical measures to evaluate the correlation between the predictors and the target feature without

the involvement of the learning algorithms. The strength of the correlation between the predictors and the target variable is then used to rank the features and thus determines their importance. On the other hand, the wrapper feature selection methods rely on the learning algorithms to determine the best subset of features that maximize the classification accuracy of the learning algorithm. Basically, wrapper methods involve using a learning algorithm, treating the selection of feature subset as a search problem, and evaluating multiple models to identify the feature subset that maximizes the performance of the model. While the wrapper methods often result in a feature subset that leads to better model performance, they are very computationally expensive (Kuhn & Johnson, 2013).

As a last step in the data preprocessing phase, the datasets were carefully analyzed to identify features that have a greater impact on the output variable. Filter feature selection methods were utilized to perform this analysis since they are faster and computationally more efficient than wrapper-based methods. The *sbf* (Selection By Filter) function in the *caret* package provides a simple interface that can be used to screen the predictors and select the optimal feature subset based on univariate statistical methods. The syntax for the *sbf* function is:

```
> library(caret)
> filterCtrl <- sbfControl(functions = caretSBF, method = "repeatedcv",
                           repeats = 10)
> filteredFeatures <- sbf(x = predictors, y = outcome,
                          sbfControl = filterCtrl)
> predictors(filteredFeatures)
```

The *sbf* control object is used to specify a list of options that can be used with *sbf*; *x* is used to specify the features to be screened; and *y* is used to specify the target variable. Finally, the best subset of features can be obtained by using the *predictors* function.

Step 3: Data splitting

After completing the preprocessing task, the datasets were split into two datasets, training and test sets. The training sets were used to construct the models, while the test set were used to evaluate the performance of the models. The *createDataPartition* function in the *caret* package was used to create the training and test sets. This function uses a stratified random sampling technique that preserves the overall class distribution of the target variable in the full datasets so that the created partitions can have a similar proportion of each class as the full dataset. In this phase, 75% of the data was allocated to the training set, and the remaining 25% was allocated to the test set. The syntax for the *createDataPartition* function is shown in the code below, where the target class variable and a parameter *p*, which defines the percentage of examples to be included in the training set, must be specified.

```
> library(caret)
> StudentsData <- read.csv("StudentsData.csv")
> split_data <- createDataPartition(StudentsData$Class, p = 0.75)
> trainingSet <- StudentsData[split_data, ]
> testSet <- StudentsData[-split_data, ]
```

Finally, using the training set, five repetitions of 10-fold cross–validation resampling technique were used to train, tune, and validate the models.

Step 4: Models Training and Tuning

In this phase, multiple models including single classifiers (i.e., LR, DT, NB, and ANN) and ensemble classifiers (i.e., bagged and boosted models) were fit to the training set. For each model, a set of hyperparameters were optimized to identify the best model fit. Using resampling methods, five repetitions of 10-fold cross–validation were used for

the optimization of the algorithms' hyperparameters. For each of the five repeats, the following steps were performed to fine tune each model:

1. Split the training set randomly into 10 folds of equal size.
2. Select a set of values for the hyperparameters.
3. Fit the model using the selected values of hyperparameters on 9 folds.
4. Evaluate the fitted model on the hold-out fold.
5. Repeat steps 3 and 4 ten times using the selected values of hyperparameters and a different fold each time as a hold-out fold.
6. Calculate the average performance across the 10 hold-out folds.
7. Repeat steps 2 to 6 for each set of hyperparameters' values.
8. Identify the optimal hyperparameters set.

Once the optimal hyperparameters values were determined, the final model was refit using the optimal hyperparameters values on the full training set.

A hyperparameter is a tuning parameter of an ML algorithm (i.e., the number of hidden layers in the ANN algorithm) which should be defined before fitting the model since its value cannot be directly learned from the data. The optimal values of these hyperparameters vary from one dataset to another and from one problem domain to another. Therefore, for each dataset, one should explore a variety of values for these hyperparameters in order to determine the optimal set of hyperparameters' values that optimize the model performance for a specific dataset. A typical approach to finding the optimal values for the hyperparameters is by using a grid search approach. In grid search, a manual set of values for each hyperparameter is specified, then models are fit and

evaluated with every combination of the hyperparameters' values to find the best combination that optimizes the model performance. Another alternative to using a grid search is to use a random search approach. In random search, random combinations of hyperparameter values are used to fit and evaluate the models. The random search approach has shown to be more computationally efficient than the grid search approach and was able to find models with equal or better performance than the ones found with the grid search approach (Bergstra & Bengio, 2012). Thus, in this study, random search of the hyperparameters' values was used to fine tune the models.

The *caret* package contains a *train* function that provides a unified interface that standardizes the model training and tuning processes. The *train* function can be used to perform a random hyperparameters search, assess the effect of tuning parameters on the model performance, find the best model across different hyperparameters, optimize custom performance metrics, and facilitate parallel processing. The syntax of the *train* function that was used to fit the models includes the following arguments:

- **x:** An object that represent the predictors.
- **y:** Represent the target variable.
- **method:** A string that specifies which classification algorithm to use.
- **preprocess:** A string vector that define the preprocessing tasks to be performed. For most of the algorithms center and scale were used.
- **metric:** A string that specifies which metric will be used to select the optimal model. In this study, the AUC was used to select the optimal model. Further explanation on AUC is provided in the evaluation step section.

- **trControl:** A list of values that defines the control parameter for the *train* function. The resampling type, the number of resampling iterations, and the type of hyperparameters search will be set using this list.
- **tuneLength:** An integer that specifies the maximum number of values that will be generated by the random search for each tuning parameter.

To ensure that the same resampled training sets are used consistently across all models, the seed for the random number generator was set before fitting each model so that the results can be reproduced and compared. The basic syntax for fitting the models is shown below:

```
## A train control object that was used by all train functions
> fitControl <- trainControl(## 10-fold Cross-Validation
                             method = "repeatedcv",
                             number = 10,
                             ## Number of resampling iterations
                             repeats = 5,
                             ## Hyperparameters search method
                             search = "random",
                             ## A caret function that calculates
                             ## the AUC, sensitivity and specificity
                             summaryFunction = twoClassSummary,
                             ## Predict the class probabilities
                             ## to calculate the ROC curve
                             classProbs = TRUE,
                             ## Allow parallel processing
                             allowParallel = TRUE)

## For each algorithm, two models were fit.
## Fitting a model based on all features in the dataset


> set.seed(2018)
> modelFit_All <- train(x = fullFeatures,
                        y = trainingSet$Class,
                        method = "AlgorithmName",
                        trControl = fitControl,
                        preProc = c("center", "scale"),
                        tuneLength = 30,
                        metric = "ROC")
## Predicting test data using the final model
> modelFit_All_Predicition <- predict(modelFit_All, newdata = testSet)

## Fitting a model based on filtered features set.
```

```
> set.seed(2018)
> modelFit_Filtered <- train(x = filteredFeautures,
                             y = trainingSet$Class,
                             method = "AlgorithmName",
                             trControl = fitControl,
                             preProc = c("center", "scale"),
                             tuneLength = 30,
                             metric = "ROC")
## Predicting test data using the final model
> modelFit_Filtered_Predicition <- predict(modelFit_Filtered,
                                           newdata = testSet)
```

Step 4a: Application of Single Classifiers

In this study, four popular classification methods (i.e., Logistic Regression, Decision Trees, Naïve Bayes, and Artificial Neural Networks) were used to identify at-risk students. These classification methods have been chosen due to their superior performance in classifying students based on their final grades as discussed earlier in the literature review section. R implementation of these techniques was used for executing all of the experiments. A total of four experiments were conducted based on the data collection times. In the first experiment, the classification algorithms were trained and tested based on data available before the beginning of the semester. In the second experiment, the classification algorithms were trained and tested based on four-week data. Similarly, in the third and fourth experiments, the classification algorithms were trained and tested based on eight-week and 12-week data.

The following subsections discuss the ML algorithms, their tuning parameters, and the packages that were used to represent them.

Logistic Regression

The *glm* function in the R *stats* package is regularly used to fit logistic regression models. The *train* function offers an interface to the *glm* function. The following code was used to fit the LR models:

```
## Fitting a LR model based on all features in the dataset
> set.seed(2018)
> lrFit_All <- train(x = fullpredictors,
                     y = trainingSet$Class,
                     method = "glm",
                     trControl = fitControl,
                     preProc = c("center", "scale"),
                     metric = "ROC")

## Predicting test data using the final model
> lrFit_All_Predicition <- predict(lrFit_All, newdata = testSet)

## Fitting a LR model based on filtered features set.
> set.seed(2018)
> lrFit_Filtered <- train(x = filteredpredictors,
                          y = trainingSet$Class,
                          method = "glm",
                          trControl = fitControl,
                          preProc = c("center", "scale",
                                      "zv", "YeoJohnson" ),
                          metric = "ROC")

## Predicting test data using the final model
> lrFit_Filtered_Predicition <- predict(lrFit_Filtered, newdata = testSet)
```

Since the *glm* function has no tuning parameters, the *tuneLength* argument was not specified in the code used to fit the LR models.

Decision Trees

While there are many R packages with implementations of DTs models, the C5.0 package (Kuhn & Quinlan, 2017) was used to build the single decision trees. As with the other models, the *train* function provides an interface to the C5.0 algorithm that can be used to tune the models with a specified performance metric. The following code was used to build the single trees:

```
## Fitting a DT model based on all features in the dataset
> set.seed(2018)
> dtFit_All <- train(x = fullFeatures,
                     y = trainingSet$Class,
                     method = "C5.0Tree",
                     trControl = fitControl,
                     preProc = c("center", "scale"),
                     metric = "ROC")
## Predicting test data using the final model
> dtFit_All_Predicition <- predict(dtFit_All, newdata = testSet)

## Fitting a DT model based on filtered features set.
> set.seed(2018)
> dtFit_Filtered <- train(x = filteredFeautures,
                          y = trainingSet$Class,
                          method = "C5.0Tree",
                          trControl = fitControl,
                          preProc = c("zv"),
                          metric = "ROC")
## Predicting test data using the final model
> dtFit_Filtered_Predicition <- predict(dtFit_Filtered, newdata = testSet)
```

By default, the *train* function provides capability for tuning two tuning parameters of the C5.0 algorithm: trials and winnow. Trials is an integer that specifies the number of boosting iterations with a default value of 1; this value represents a single tree model. Winnow is a logical that indicates if feature selection should be used with "false" as a default value. Since the trials tuning parameter is not relevant to building a single tree, and feature selection is performed at a prior stage, the *"C5.0Tree"* method in the *train* function was used to build a single tree using the C5.0 algorithm implementation without any tuning parameters.

Naïve Bayes

The *klar* R package (Weihs, Ligges, Luebke, & Raabe, 2005) was used to fit the NB models. In the *train* function, the method values of "nb" was used to train and fine tune the NB models with two hypermeters to tune: Laplace and "usekernel". Laplace is a

numeric tuning parameter that defines the value to be used for Laplace correction with a default value of 0, which means no Laplace correction will be used. The usekernel is a logical tuning parameter that when set to TRUE, a kernel density estimate would be used to estimate the densities of continuous predictors. The code to fit the NB models was similar to the one used to fit the previous models, with value of "nb" used as a *train* method.

Artificial Neural Networks

The *nnet* R package (Venables & Ripley, 2002) was used with the *caret* package to train and fine tune the ANN models. The code to fit the ANN models was similar to the one used to fit the other models, with a *train* method value of "nnet". The *train* function provides ability to tune two hyperparameters of the *nnet* model: size and decay. The size is a numeric tuning parameter that specifies the number of units in the hidden layer. The decay is also a numeric tuning parameter that represents a regularization term that is used to decay the weights in proportion to their size.

Step 4b: Application of Ensemble Classifiers

Ensemble techniques can be used to improve the prediction accuracy of base classifiers (Rokach, 2010). This study employed two common ensemble techniques: bagging and boosting. Similar to the previous step, ensemble classifiers were trained and tested four times based on the data collection time.

Bagging

Random Forest (RF) algorithm was used to represent the bagging techniques. The RandomForest R package (Liaw & Wiener, 2002) was used with the caret package to

train and fine tune the RF models. The code to fit the RF models was similar to the one used to fit the other models, with a train method value of "rf". The train function provides ability to tune one hyperparameters of the rf model: mtry. The mtry tuning parameter refers to the number of predictors randomly selected at each split.

Boosting

The Extreme Gradient Boosting (XGBoost) classifier was used to represent the booting techniques. The tuning process was similar to the one used for tuning the single classifiers. Each boosted model was automatically tuned and evaluated using five repetitions of the 10-fold cross-validation resembling method. Table 4 below provides a list of the boosting algorithm, its R package, and its hyperparameters.

Table 4. Boosting Algorithms List

Boosting Algorithm	**R Package**	**Method Name** (for *train* function)	**Hyperparameters**
Extreme Gradient Boosting	xgboost (Chen, He, Benesty, Khotilovich, & Tang, 2018)	xgbTree	- Number of Boosting Iterations - Max Tree Depth - Shrinkage - Min. Loss Reduction - Subsample Ratio of Columns - Min. Sum of Instance Weight - Subsample Percentage

Step 5: Evaluation of the Results

After completing the fourth step, an evaluation of the overall classification performance was performed. The evaluation of classification performance of each ML

technique was based on three criteria: sensitivity, specificity, and AUC. The *twoClassSummary* function in the *caret* package was used to generate the evaluation results. The measurement of these performance criteria is based on the classification's confusion matrix, which is shown in Table 5 below.

Table 5. Classification Confusion Matrix

		Predicted Value	
		Positive	Negative
Actual Value	Positive	**TP** True Positive	**FN** False Negative
	Negative	**FP** False Positive	**TN** True Negative

Table 5 shows the confusion matrix for general "positive" and "negative" classes. In this research, the positive class represented the "at-risk" class, while the negative class represented the "good-standing" class. As the confusion matrix shows, the predictions can fall into one of the following four groups:

- True Positive (TP): Correctly classified as "at-risk".
- True Negative (TN): Correctly classified as "good-standing".
- False Positive (FP): Incorrectly classified as "at-risk".
- False Negative (FN): Incorrectly classified as "good-standing".

Using the confusion matrix, the sensitivity and specificity of a model can be calculated. The sensitivity of a model (also known as true positive rate) measures the proportion of positive (at-risk) examples that were correctly classified as such. As

shown in Equation 3.2, sensitivity is calculated by dividing the total number of true positives by the total number of positive examples:

$$Sensitivity = \frac{TP}{TP + FN}. \quad (3.2)$$

The specificity of a model (also known as true negative rate) measures the proportion of negative (good-standing) examples that were correctly classified as such. It's calculated by dividing the total number of true negatives by the total number of negative examples:

$$Specificity = \frac{TN}{TN + FP}. \quad (3.3)$$

The optimal goal is to maximize both of these performance measures since sensitivity only measures the classifiers' performance in predicting the positive (at-risk) class, while specificity only measures the classifiers' performance in predicting the negative (good-standing) class. However, there is often a trade-off between these performance measures since an increase in one measure usually results in a decrease in the other. One way to combine sensitivity and specificity into a single performance measure is by using the receiver operating characteristic (ROC) curve. The ROC is used to evaluate the trade-off between sensitivity and false positive rate (1 – specificity) by plotting the resulting sensitivity against the false positive rate for a range of thresholds. The performance of a classifier is summarized over the plotted thresholds and can be quantified by the area under the ROC curve (AUC). The AUC value ranges between 0.5 to 1, where 1 represents a perfect classifier (James et al., 2013; Kuhn & Johnson, 2013). Therefore, in this study, the AUC was used as a performance metric to be optimized by the learning algorithms.

Summary

To answer the study's first question, a comparison of the classification performance for all predictive models in each experiment was conducted to identify the best ML techniques for classifying at-risk students. The second question pertains to the feasibility of using single classifiers to generate early predictions of at-risk students and was answered by comparing the classification performance of single classifiers trained in all of the four experiments. Also, the classification performance of single and ensemble classifiers in all experiments was compared to answer the third question. The fourth question was answered by identifying the best classifier in each experiment and then comparing their classification performance to observe how the classification accuracy improved over time. The fifth question of this study is concerned with the generalizability of the predictive models across on-campus and online courses. To answer this question, two sets of data were used to evaluate the predictive models. The first dataset was based on on-campus students, while the other dataset was based on online students only. The classification performance for all ML techniques tested on each dataset was compared and reported. Finally, to answer the sixth question, the importance of features for the ML technique in all experiments were generated and presented.

Chapter 4

Results

Overview

In this chapter, the results of the classifiers performance in the four experiments are presented. The results are discussed in two main sections: evaluation of classifiers and comparison of classifiers. In the first section, the results of each classifier trained on different datasets will be described and compared individually. For each classifier, the performance across different tuning parameters, training and testing performance, and variables importance will be discussed. However, not all classifiers provide the capability to estimate the importance of variables. Hence, variable importance discussions and plots will only be included if the classifier can estimate the importance of variables. In the second section, the performance of all classifiers trained on the same dataset will be described and compared. Each experiment represents a single dataset that corresponds to a point of time in the semester. Prior to the results discussion, a short discussion of datasets, data preprocessing, and data splitting is provided.

The students' data used in this research was obtained from Jazan University, Saudi Arabia. The sample of this study was drawn from courses that made significant use of the LMS. Initially, all classes offered during Fall 2017, Spring 2018, and Fall 2018 were considered for this study. Then, a report that included details about the courses (sections) usage of LMS was generated. Next, the courses were selected based on their usage level of LMS. All courses that had used LMS for at least 15 different activities were included in this study, which resulted in a total of 2,483 sections/groups. Once the

list of courses had been prepared, all data pertaining to these courses were obtained including data about students from the SIS and their online activities from the LMS. Additionally, data about the courses were obtained from SIS and LMS. As a result, the initial datasets included a total of 159,535 data points. Table 6 shows the total observations per semester and the distribution of At-Risk and Good-Standing classes in the datasets.

Table 6. Distribution of Classes Across Semesters

	Semester	At-Risk	Good-Standing	Total
1	Fall 2017	15,405	35,976	51,381
2	Spring 2018	13,936	39,900	53,836
3	Fall 2018	14,741	39,577	54,318

During the initial check of the data sets, the data type of each variable was inspected to ensure that it represents the correct data type. Next, missing values analysis was conducted and as a result, a total of 1,167 cases were removed from the datasets which left us with a total of 158,368 data points at the end of the missing values treatment phase. Furthermore, additional preprocessing procedures were conducted for some of the classifiers. Details of theses preprocessing tasks will be provided in the models' results section. Additionally, the response variable AT_RISK was created based on the conversion of the STU_GRADE variable to a factor with two levels: YES (for At-Risk students) and NO (for Good-Standing students). The assignment of students into these levels was based on the students' final grades where a student with grade lower than 70 was considered as At-Risk.

As part of the preprocessing step, some variables were used to derive new variables. Table 7 presents the variables derived based on variables from the SIS dataset.

Table 7. Variables Derived Based on SIS Dataset

Variable Name	Description
STU_AGE	The students age was calculated using the STU_BIRTH_YEAR variable as follow: - For 20171 semester, AGE = 2016 - STU_BIRTH_YEAR. - For 20172 semester, AGE = 2017 - STU_BIRTH_YEAR. - For 20181 semester, AGE = 2017 - STU_BIRTH_YEAR.
STU_HS_UNIV_YEARS	The years students spent between graduating from high school and joining the university.
STU_CHG_COLLEGE	Whether a student had transferred from the college that was admitted into.
STU_CHG_MAJOR	Whether a student had changed the major that he was initially admitted into.
STU_cGPA_DIF	The difference in the students' cumulative GPA over the past two semesters.
STU_GPA_DIF	the difference in the students' GPA over the past two semesters.
LVL_DIF	the difference in the students' levels and the courses level.
STU_TOTAL_FLD_CRD	The number of credits the students failed during their studies in the university.
STU_CRS_COLLEGE	Whether the course was taught by faculty from the same college the students enrolled in or belong to another college and is part of the university requirement courses.

Additionally, the LMS variables were used to derive new variables including the summed attributes for weeks 5-8 and the summed attributes for weeks 9-12. Moreover, numerous percentage features were created for each activity by dividing the number of submitted activities by the total number of assigned activities. At the end of the preprocessing phase, the features were evaluated using feature selection methods and as a

result a filtered features set was created for each dataset. Table 8 shows the datasets and the number of features in the full features set and filtered features set. For each experiment, two datasets were created, one with the full features set while the other only included the features selected by the features-selection methods.

Table 8. Number of Observations and Features in the Data Sets

	Dataset	Description	Features Set	Number of	
				Features	Observations
1	Week0	This dataset only contains historical students' data that was available before the beginning of the class	Full Set	46	157,227
2			Filtered Set	35	
3	Week4	This dataset contains features available at week 0 in addition to the available data by the end of week 4	Full Set	98	157,227
4			Filtered Set	35	
5	Week8	This dataset contains data from previous weeks in addition to data available by the end of week 8	Full Set	239	157,227
6			Filtered Set	120	
7	Week12	This dataset contains data from previous weeks in addition to data available by the end of week 12	Full Set	427	157,227
8			Filtered Set	150	

Once the preprocessing tasks were completed, stratified random sampling was used to split the datasets into training and test sets where 75% of the data was allocated to the training set, and the remaining 25% was kept for the evaluation of the trained models. Additionally, resampling methods were used to increase the reliability of the trained

models. As a result, five repetitions of a 10-fold cross-validation technique were used to validate the models during the training phase. Hence, the training performance results are an average of 50 runs.

Evaluation of Classifiers

For each of the classifiers evaluated, the classifier's training and evaluation results are presented. The results of training performance are an average of 50 models trained on 50 different folds, while the evaluation results are based on the hold out test set. Statistics on the classifications' performance (i.e. AUC, sensitivity, specificity) for each classifier and dataset are plotted and summarized in tables. Furthermore, the classifiers training and evaluation performance is compared and discussed. Finally, variables importance discussions and plots are provided for each of the evaluated classifiers.

Logistic Regression

The logistic regression classifier was trained on eight distinct datasets. For each of the four experiments (i.e., Week 0, Week 4, Week 8, and Week 12), the LR classifier was trained on two datasets. The first dataset contained the full features set, while the second dataset only included the filtered features set. Five repetitions of a 10-fold cross-validation procedure were used to validate the trained models. Then, the models trained on the full training set were evaluated using the hold out set test. The AUC measure was used to evaluate and rank the models. The AUC and ROC terms will be used interchangeably throughout the results section and refer to the area under the ROC curve. Prior to training the models, the categorical variables were converted to dummy variables first. For each categorical variable, each group gets its own dummy variable with value of

one or zero, which indicates the presence of the category. Then, the input variables were transformed using centering and scaling preprocessing procedures.

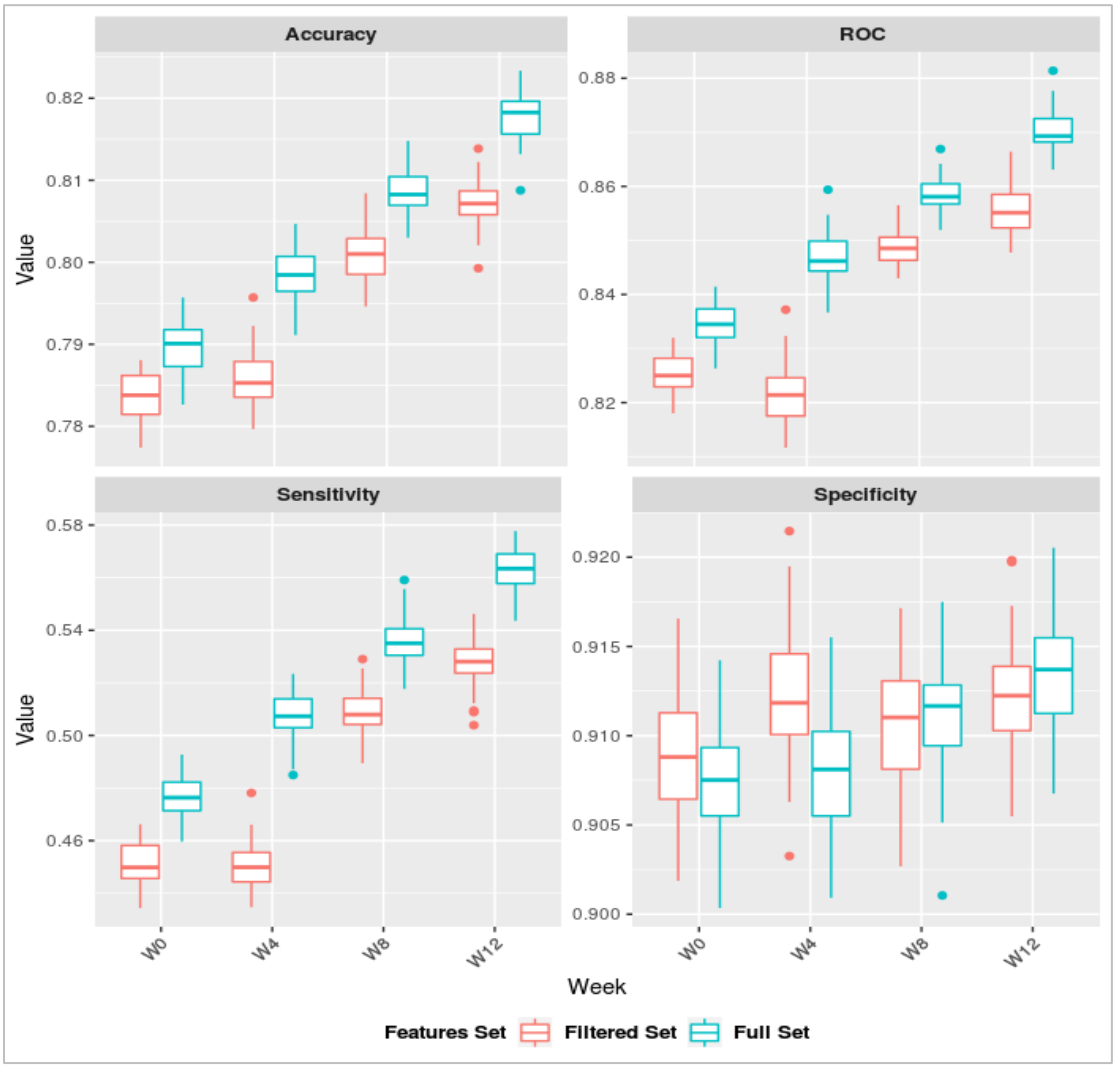

Figure 3. Box-Whisker Plots for the Resampling Distributions of LR Models

Figure 3 shows a box-whisker plot for the distributions of training performance measures across the 50 models trained for each dataset. The box-whisker plot compactly displays the distribution of the performance measures. The figure shows the full features datasets in blue with the filtered features datasets in red. It can be seen from the plot that the performance of models trained on the full features' sets outperformed the performance of the models trained on the filtered features set in all experiments.

Moreover, the performance of models trained on the full features sets shows consistent improvement as the course progresses throughout the semester. Table 9 provides a summary statistic of the training performance across all datasets. Considering AUC as the primary performance criterion, the full features datasets resulted in the best performance in training the LR models. Hence, all further discussion and analysis will only pertain to models trained on the full features sets since their performance is superior to those trained on the filtered features sets.

Table 9. Logistic Regression Models Training Summary Statistics

Metric	Week	Features Set	Mean	Min	Max	SD
AUC	W0	Full Set	0.8346	0.8263	0.8414	0.0035
		Filtered Set	0.8252	0.8181	0.8320	0.0034
	W4	Full Set	0.8467	0.8367	0.8594	0.0045
		Filtered Set	0.8215	0.8117	0.8372	0.0052
	W8	Full Set	0.8586	0.8519	0.8669	0.0030
		Filtered Set	0.8486	0.8430	0.8565	0.0033
	W12	Full Set	0.8701	0.8631	0.8814	0.0033
		Filtered Set	0.8556	0.8478	0.8664	0.0039
Sensitivity	W0	Full Set	0.4764	0.4597	0.4927	0.0080
		Filtered Set	0.4511	0.4344	0.4662	0.0076
	W4	Full Set	0.5074	0.4850	0.5234	0.0087
		Filtered Set	0.4502	0.4347	0.4781	0.0089
	W8	Full Set	0.5359	0.5177	0.5591	0.0084
		Filtered Set	0.5090	0.4894	0.5290	0.0082
	W12	Full Set	0.5633	0.5436	0.5777	0.0079
		Filtered Set	0.5274	0.5039	0.5462	0.0088
Specificity	W0	Full Set	0.9074	0.9004	0.9142	0.0030
		Filtered Set	0.9087	0.9019	0.9166	0.0033
	W4	Full Set	0.9080	0.9009	0.9155	0.0034
		Filtered Set	0.9121	0.9033	0.9215	0.0035
	W8	Full Set	0.9110	0.9011	0.9175	0.0032
		Filtered Set	0.9108	0.9027	0.9171	0.0034
	W12	Full Set	0.9134	0.9068	0.9205	0.0030
		Filtered Set	0.9122	0.9055	0.9198	0.0029

Table 10. Logistic Regression Models Performance Summary

Week	Features Set	AUC		Sensitivity		Specificity	
		Train	Test	Train	Test	Train	Test
W0	Full Set	0.8346	0.8323	0.4764	0.4758	0.9074	0.9064
W4	Full Set	0.8467	0.8486	0.5074	0.5092	0.9080	0.9080
W8	Full Set	0.8586	0.8595	0.5359	0.5414	0.9110	0.9110
W12	Full Set	0.8701	0.8682	0.5633	0.5692	0.9134	0.9114

Tables 10 provides summary statistics for the training and evaluation performance metrics for models trained on the full features' sets. The results of the evaluation performance metrics are close to those obtained from the training stage, which shows that the implemented validation procedures were effective. Starting with week zero dataset, the best model trained on this dataset was able to achieve an AUC of 83%, a sensitivity of 47%, and a specificity of 90%. While the model trained on this dataset achieved an excellent specificity score, it fell below expectations for sensitivity. Sensitivity refers to the ability of the predictive models to accurately predict the At-Risk class, which is the interest of this research problem. However, the model trained on week zero dataset was only able to predict less than half of the total At-Risk students.

Models trained on week four, eight, and 12 datasets sustained improved scores in all performance metrics. As the semester progressed, each week's model was able to improve by 1% in AUC, 3% in sensitivity, and less than 0.5% in specificity. By week 12, the model was able to achieve about 87% in AUC, 57% in sensitivity, and 91% in specificity. Figure 4 plots the ROC curves for the training and evaluation stage.

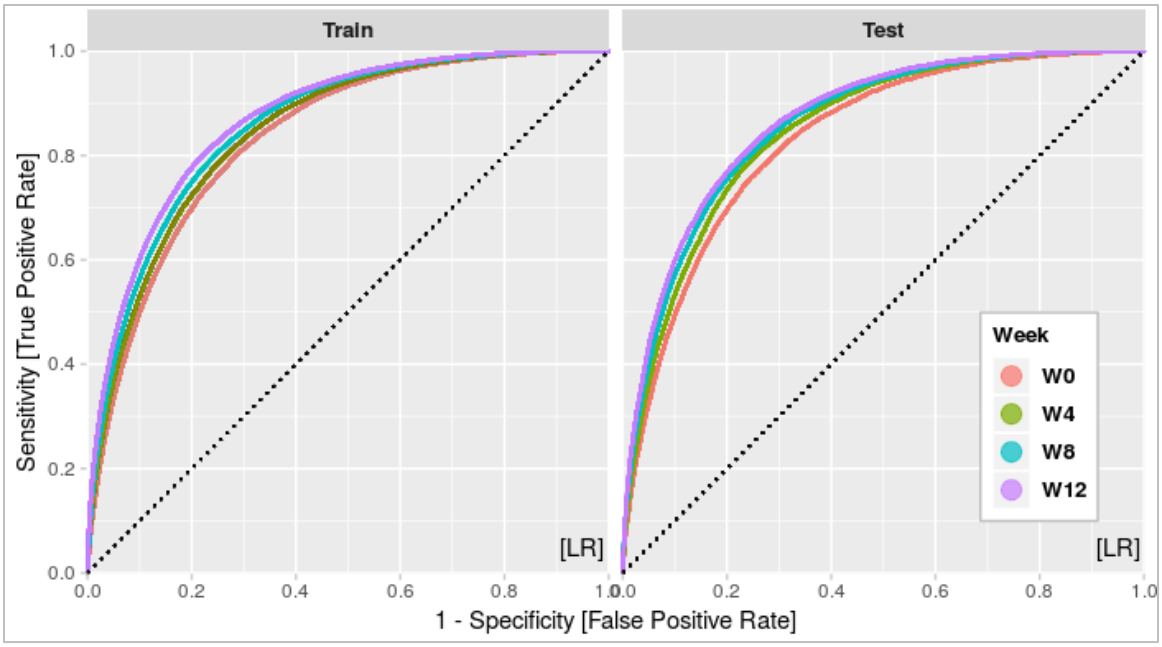

Figure 4. ROC Curves for the Logistic Regression Models

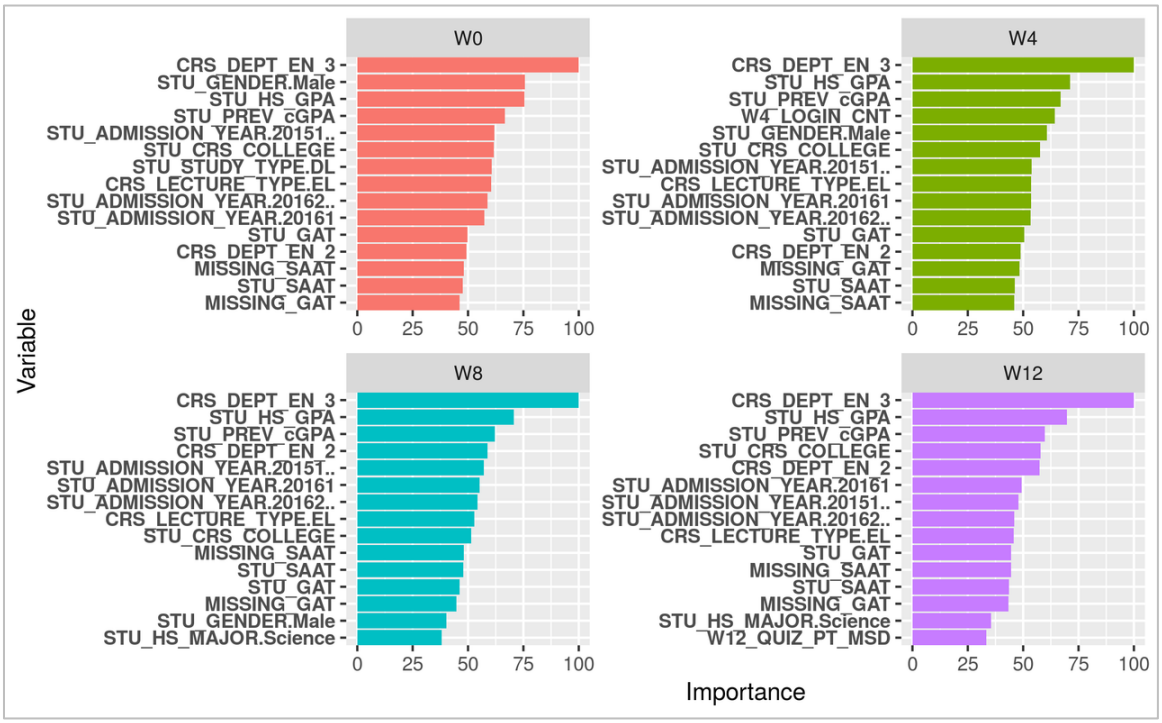

Figure 5. Variable Importance Plots for Each LR Model

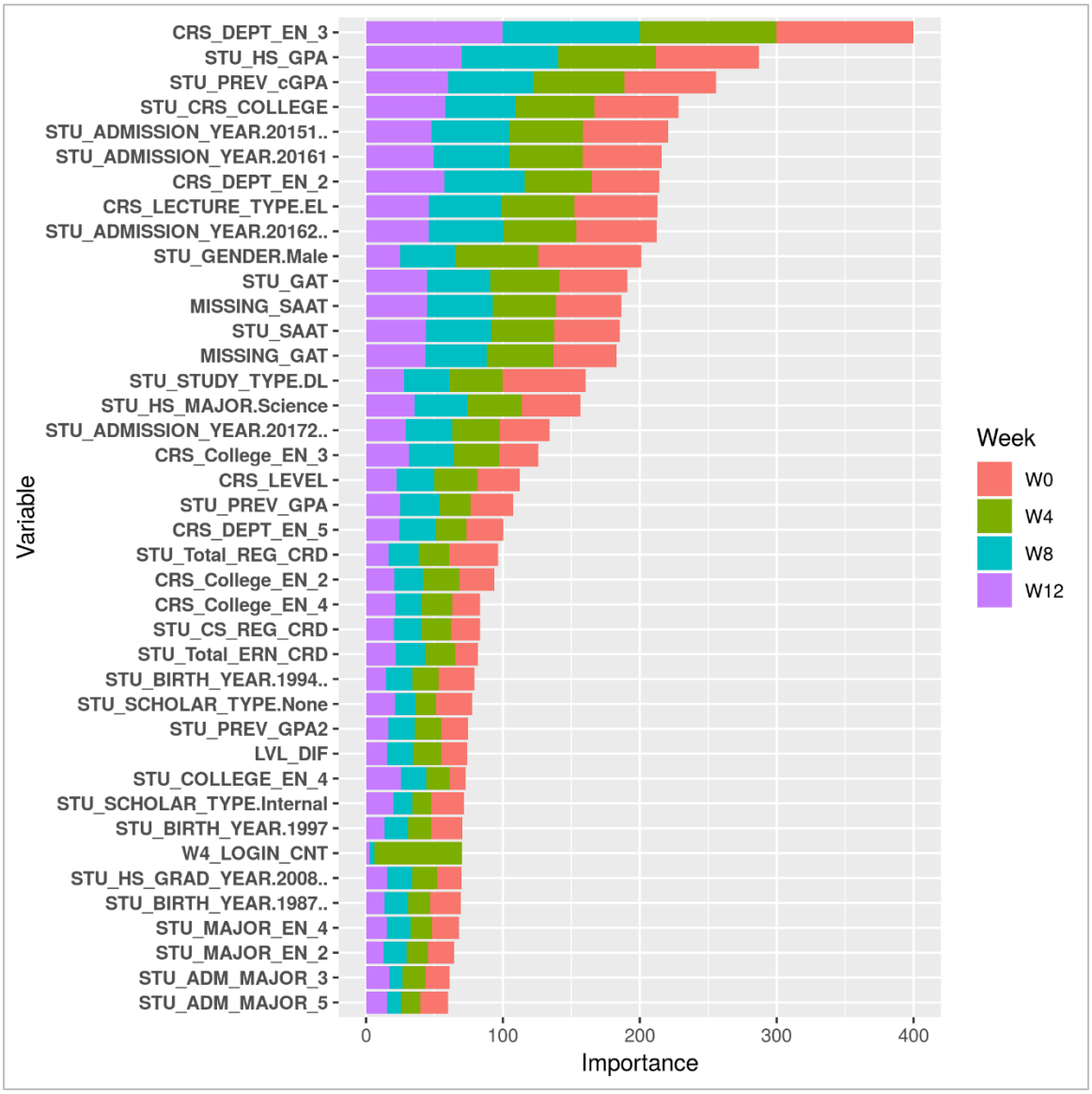

Figure 6. Overall Variables Importance Plot for the LR Models

Figure 6 plots the top 40 variables by combining their importance across all models trained on the full features' sets. The variables importance is based on their z-value, which is calculated by dividing the regression coefficient by its standard error. Most of the important variables are from the SIS dataset, which was available since week zero. Figure 5 shows a closer look at the variables' importance for each dataset

individually. In the individual variable importance plots, the top 15 variables are reported for each week's dataset. Similar to the overall importance plot, the individual plots show that the SIS variables are dominating the top 15 most important variables across all datasets.

Decision Trees

As previously mentioned, C5.0 algorithm was selected to represent decision trees classifiers. Like the LR classifier, the C5.0 classifier was trained on two datasets for each of the four experiments (i.e., Week 0, Week 4, Week 8, and Week 12). The C5.0 algorithm has two tuning parameters: trials and winnow. However, both tuning parameters were kept constant throughout the training phase since the trials tuning parameter was not relevant to building a single tree, and feature selection was performed at a prior stage. The area under the ROC curve will be used to compare and select the best performing models.

Figure 7 shows a box-whisker plot for the distribution of training performance measures across the 50 models trained for each dataset. For some performance metrics, the plot demonstrates that the performance of the model trained on the full features sets is better than the model trained on the filtered sets, while the filtered sets models performed better on other performance metrics. Considering AUC as the primary performance criterion, the full features datasets resulted in the best performance in training the C5.0 models. Hence, all further discussion and analysis will only pertain to models trained on the full features sets since their performance is superior to those trained on the filtered features sets. Additionally, the performance of models trained on the full features sets shows a slight improvement as the course progresses toward the end of the semester,

while the models trained on the filtered sets suffered a decrease in performance on week four but recovered the improvement trend by week eight. Table 11 provides more details on the distribution of the training performance across all datasets. All the performance metrics across all datasets have a close to zero standard deviation, which indicates that the value of the performance metrics for all 50 trained models were close to the reported means.

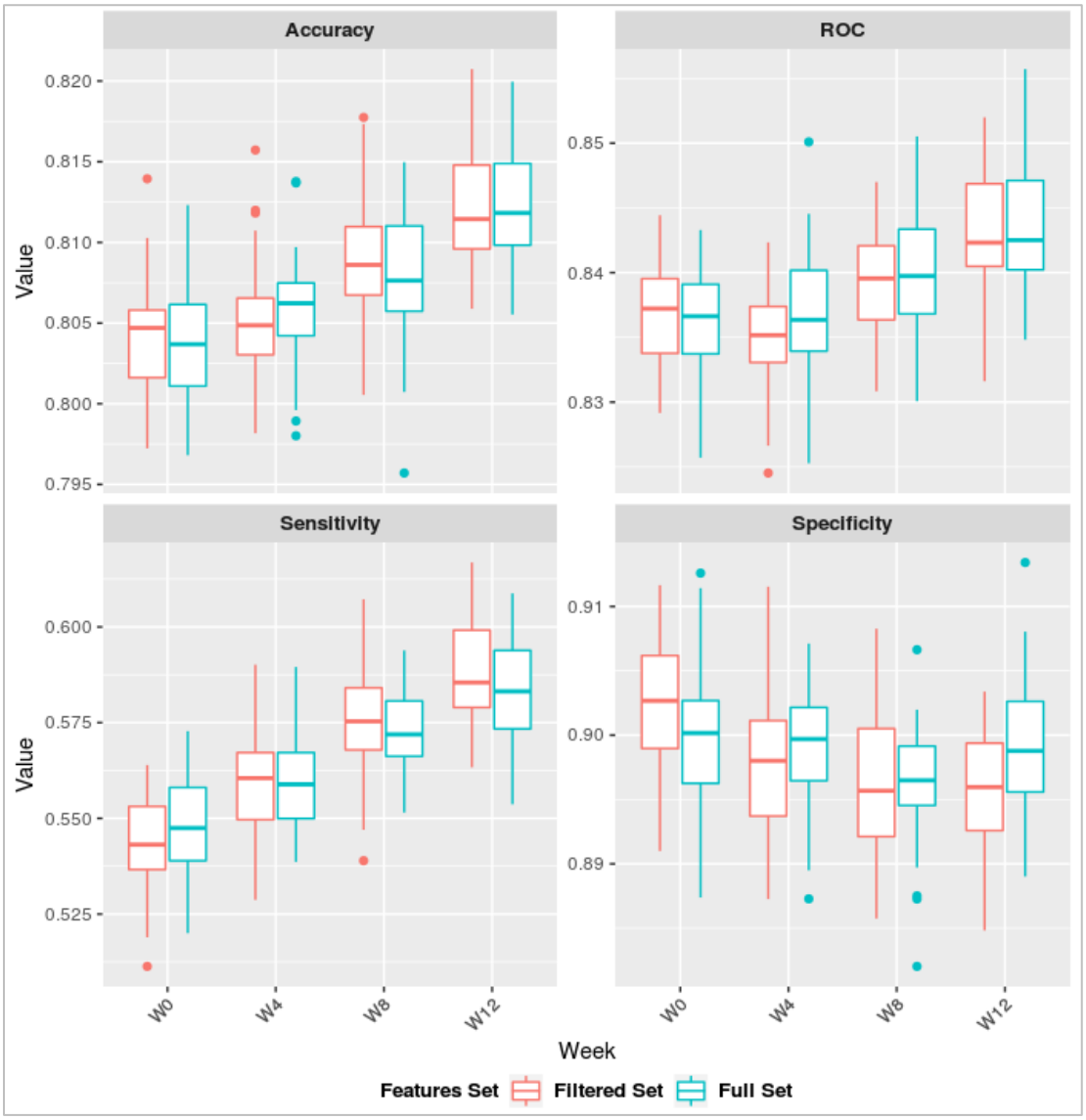

Figure 7. Box-Whisker Plots for the Resampling Distributions of LR Model

Table 11. C5.0 Models Training Summary Statistics

Metric	Week	Features Set	Mean	Min	Max	SD
AUC	W0	Full Set	0.8360	0.8257	0.8433	0.0041
		Filtered Set	0.8367	0.8292	0.8444	0.0040
	W4	Full Set	0.8366	0.8253	0.8501	0.0047
		Filtered Set	0.8353	0.8245	0.8423	0.0039
	W8	Full Set	0.8399	0.8301	0.8505	0.0046
		Filtered Set	0.8393	0.8308	0.8470	0.0035
	W12	Full Set	0.8437	0.8348	0.8557	0.0049
		Filtered Set	0.8431	0.8316	0.8520	0.0050
Sensitivity	W0	Full Set	0.5478	0.5200	0.5728	0.0134
		Filtered Set	0.5440	0.5113	0.5639	0.0117
	W4	Full Set	0.5580	0.5386	0.5895	0.0109
		Filtered Set	0.5595	0.5287	0.5901	0.0137
	W8	Full Set	0.5728	0.5515	0.5939	0.0099
		Filtered Set	0.5751	0.5389	0.6072	0.0129
	W12	Full Set	0.5832	0.5537	0.6087	0.0141
		Filtered Set	0.5888	0.5633	0.6168	0.0135
Specificity	W0	Full Set	0.8998	0.8874	0.9126	0.0057
		Filtered Set	0.9021	0.8910	0.9117	0.0057
	W4	Full Set	0.8990	0.8873	0.9071	0.0045
		Filtered Set	0.8974	0.8873	0.9115	0.0053
	W8	Full Set	0.8964	0.8820	0.9066	0.0043
		Filtered Set	0.8964	0.8858	0.9083	0.0055
	W12	Full Set	0.8987	0.8890	0.9134	0.0052
		Filtered Set	0.8959	0.8848	0.9034	0.0046

Table 12. C5.0 Models Performance Summary

Week	Features Set	AUC		Sensitivity		Specificity	
		Train	Test	Train	Test	Train	Test
W0	Full Set	0.8360	0.8327	0.5478	0.5487	0.8998	0.8981
W4	Full Set	0.8366	0.8394	0.5580	0.5660	0.8990	0.8986
W8	Full Set	0.8399	0.8433	0.5728	0.6032	0.8964	0.8885
W12	Full Set	0.8437	0.8452	0.5832	0.5799	0.8987	0.8993

Table 12 provides a summary of the training and evaluation performance metrics for models trained on the full features' sets. Again, the results of the evaluation performance metrics are close to those obtained from the training stage. Starting with week zero dataset, the best model trained on this dataset was able to achieve an AUC of 83%, a sensitivity of 55%, and a specificity of 90%. Week four results were similar to those obtained on week zero with sensitivity increasing by only about 1.5%. Overall, it seems that there is no significant improvement in performance as the semester progresses towards the end. Figure 8 plots the ROC curves for the training and evaluation stage, which shows that almost all ROC curves seem identical and hard to separate.

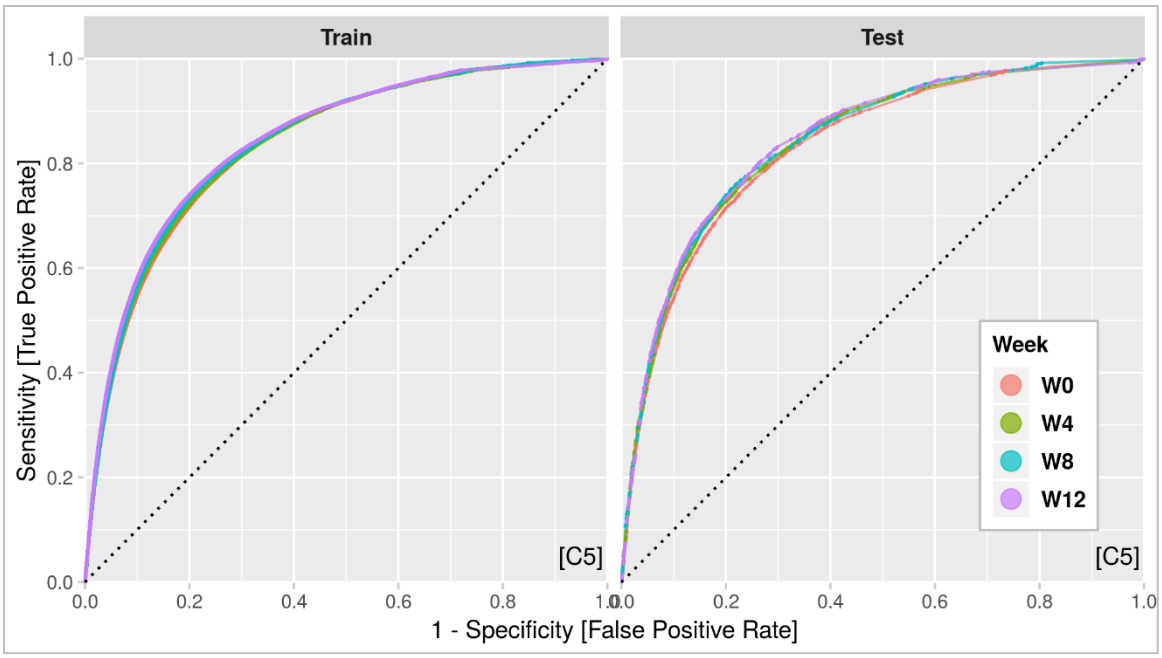

Figure 8. ROC Curves for the C5.0 Models

Figure 9 shows the top 40 variables according to their importance to the C5.0 models. The C5.0 algorithm measures variable importance by calculating the percentage of samples that fall into all leaf nodes after the split. In contrast to the LR models, the plot

shows that only half of the reported variables were from the SIS datasets, and the remaining were time dependent variables from the LMS datasets. Even though the C5.0 models seemed to better utilize the time dependent variables, LR model trained on the week 12 dataset performed better than the one trained using C5.0 on the same dataset. Figure 10 reports the importance of the top 15 variables for each week's model. A closer look at the variable importance for week 12 shows that most of the reported top 15 variables were from the LMS dataset, which may explain the difference in AUC between week 12's LR and C5.0 models.

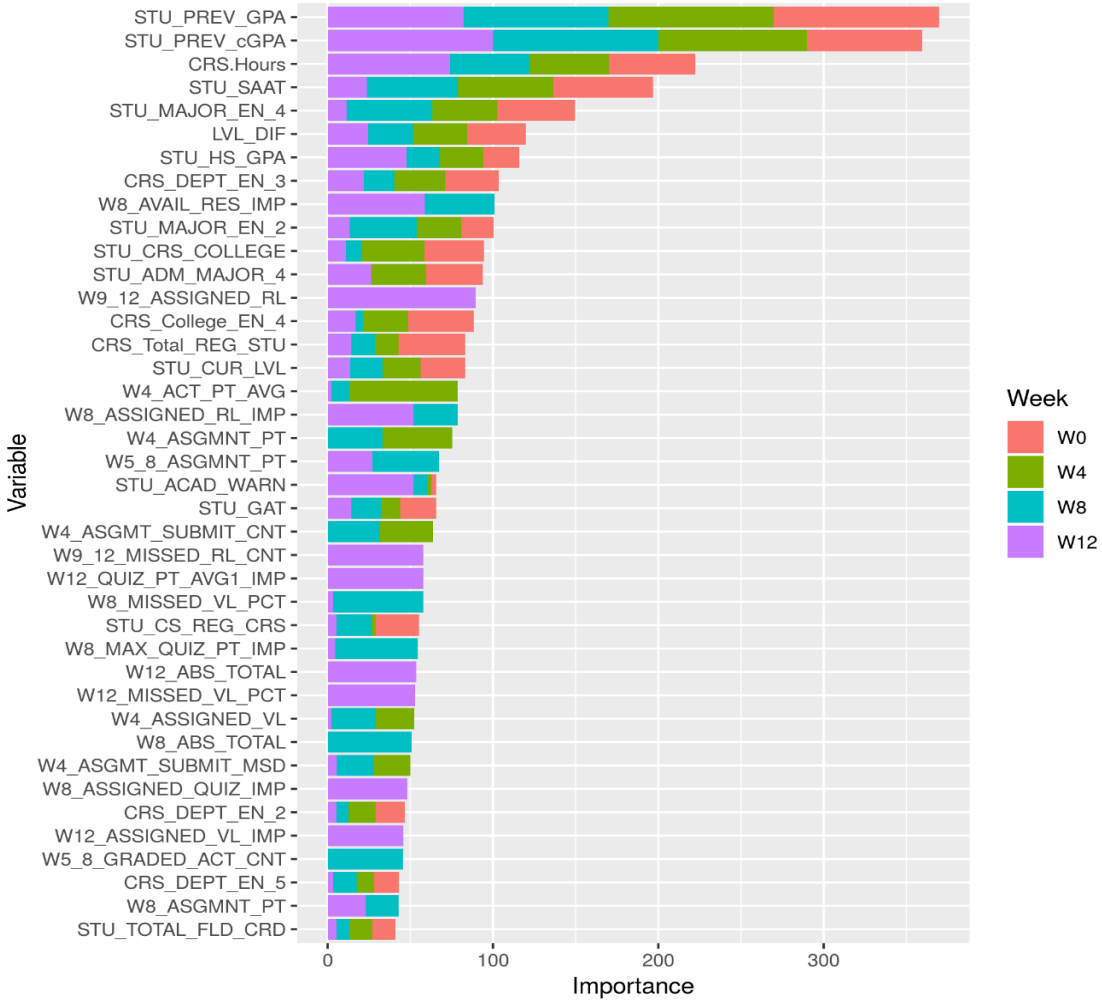

Figure 9. Overall Variables Importance Plot for the C5.0 Models

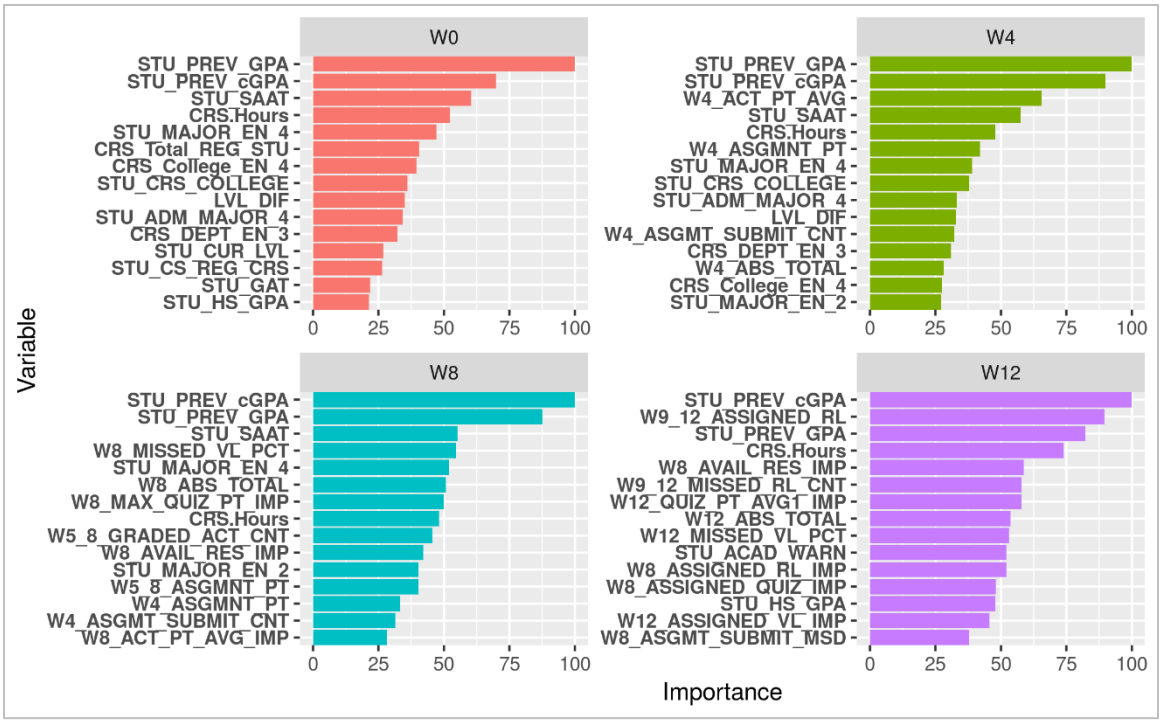

Figure 10. Variable Importance Plots for Each C5.0 Model

Naïve Bayes

Similar to the previous classifiers, Naïve Bayes classifier was trained on two datasets for each of the four experiments. As discussed in the methodology section, NB algorithm has three tuning parameters: distribution type, Laplace correction, and bandwidth adjustment. The input variables were modeled using a nonparametric density estimation and a Laplace correction ranging between 0 and 2. Table 13 lists the tuning parameters and the values tried for each one of them. The best tuning parameters that were used to train each week's final model are listed in Table 14. Figure 11 plots the average AUC associated with each of the tuning parameter values. The upwards arrows indicate the best tuning parameter value for each model, while the downwards arrows indicate the worst tuning parameter. The best and worst values for a tuning parameter are determined based on the average AUC value associated with the tuning parameter value.

Table 13. List of Naïve Bayes Algorithm Tuning Parameters

Parameter	Description	Class	Values	Count
usekernel	Distribution Type	logical	TRUE	1
fL	Laplace Correction	numeric	0, 0.5, 1, 1.5, 2	5
adjust	Bandwidth Adjustment	numeric	0.5, 1, 1.5, 2, 2.5, 3, 3.5	7

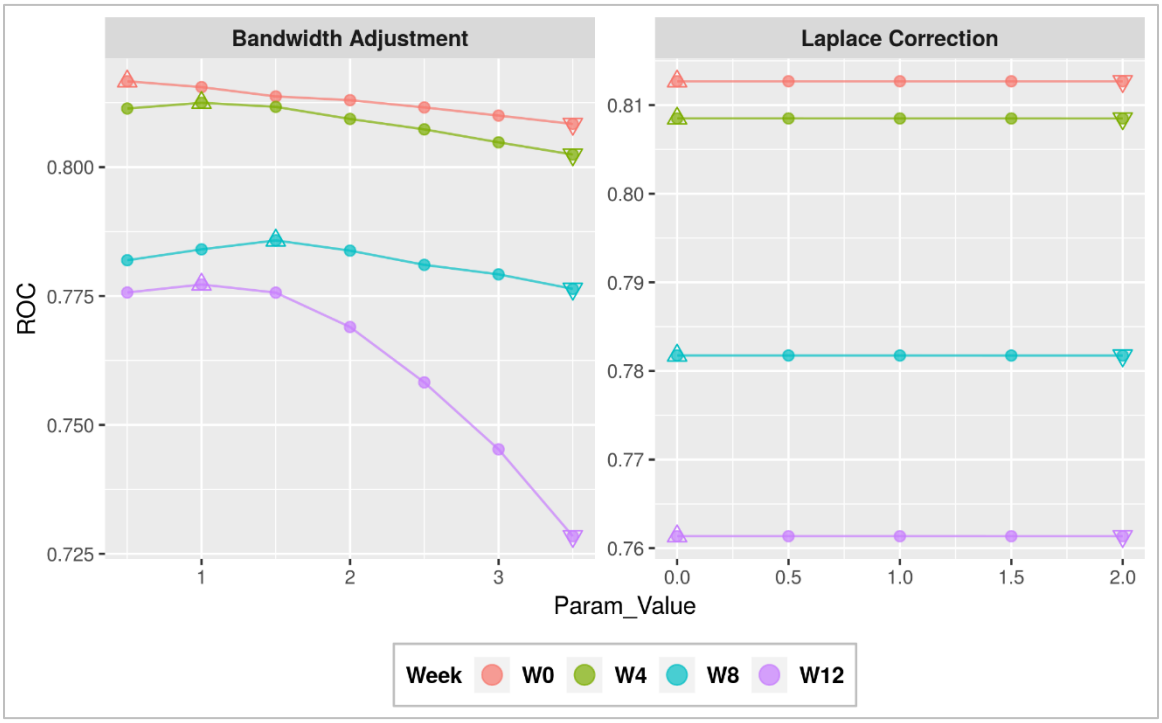

Figure 11. Relationship between NB Tuning Parameters and AUC

Table 14. Naïve Bayes Best Tuning Parameters for Each Week's Models

Week	Features Set	fL	usekernel	adjust
W0	Full Set	0	true	0.5
W0	Filtered Set	0	true	0.5
W4	Full Set	0	true	1
W4	Filtered Set	0	true	0.5
W8	Full Set	0	true	1.5
W8	Filtered Set	0	true	1.5
W12	Full Set	0	true	1
W12	Filtered Set	0	true	1.5

Figure 12. Box-Whisker Plots for the Resampling Distributions of NB Models

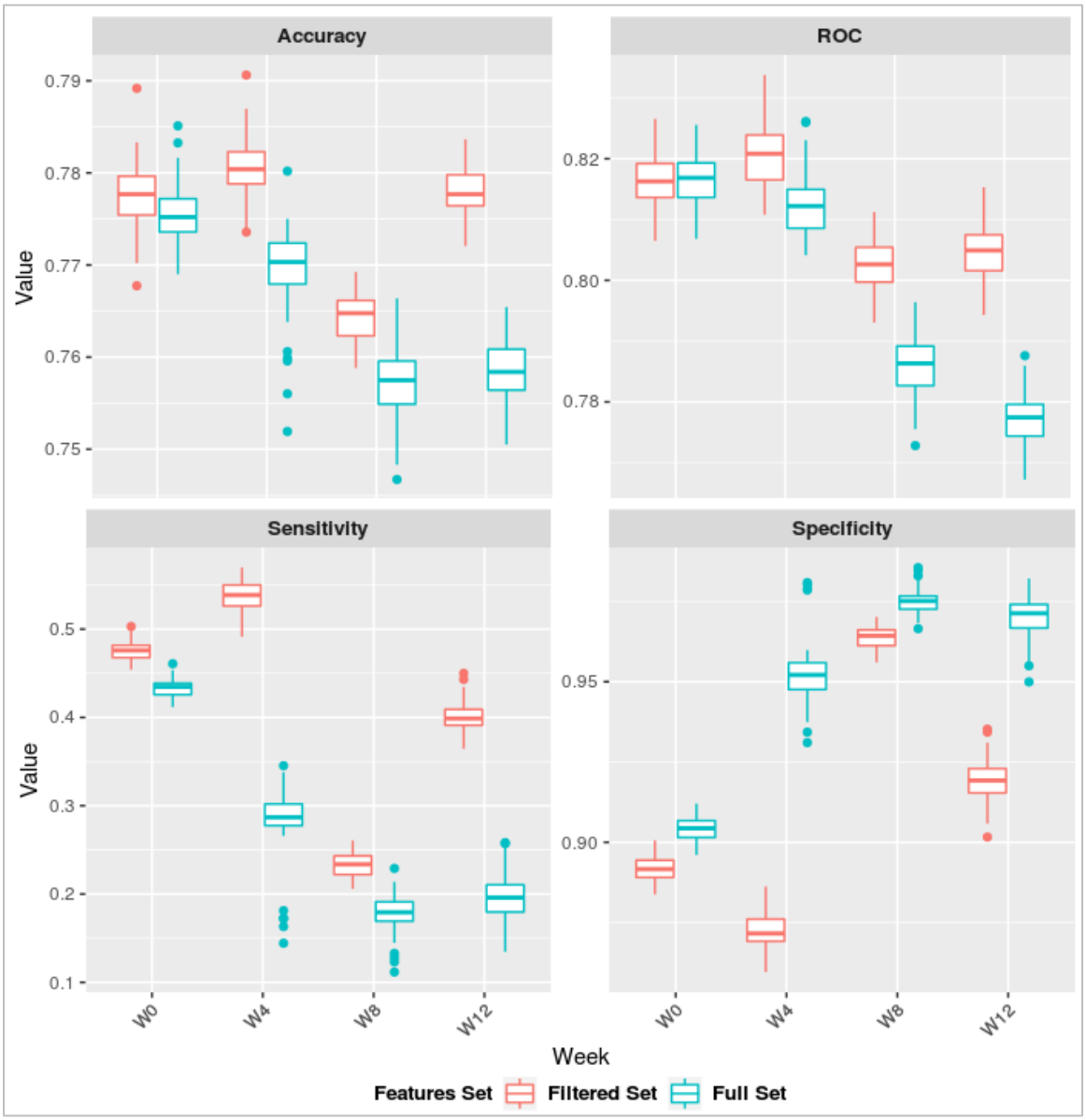

Figure 12 shows a box-whisker plot for the distribution of training performance measures across the 50 models trained for each dataset. Unlike the previous classifiers, the plot shows that the performance of the model trained on the filtered features sets is better than those trained on the full features' sets. Furthermore, the values of performance measures for models trained on the full features' sets have an unusual quantity of outliers, which may suggest instability in these models. Surprisingly, no consistent improvements

in performance throughout the semester were noted with week four having the highest AUC value and week eight having the worst AUC value. Table 15 provides more details on the distribution of the training performance across all datasets. Considering AUC as the primary performance criterion, the filtered features datasets resulted in the best performance in training the NB models. Hence, all further discussion and analysis will only pertain to models trained on the filtered features sets since their performance is superior to those trained on the full features' sets.

Table 15. Naïve Bayes Models Training Performance Summary Statistics

Metric	Week	Features Set	Mean	Min	Max	SD
AUC	W0	Full Set	0.8167	0.8068	0.8256	0.0040
		Filtered Set	0.8165	0.8065	0.8266	0.0040
	W4	Full Set	0.8125	0.8041	0.8261	0.0049
		Filtered Set	0.8205	0.8108	0.8338	0.0048
	W8	Full Set	0.7858	0.7728	0.7964	0.0051
		Filtered Set	0.8027	0.7930	0.8112	0.0043
	W12	Full Set	0.7772	0.7672	0.7876	0.0046
		Filtered Set	0.8047	0.7943	0.8153	0.0042
Sensitivity	W0	Full Set	0.4334	0.4117	0.4608	0.0101
		Filtered Set	0.4742	0.4541	0.5029	0.0115
	W4	Full Set	0.2809	0.1443	0.3453	0.0424
		Filtered Set	0.5366	0.4913	0.5697	0.0184
	W8	Full Set	0.1783	0.1117	0.2290	0.0222
		Filtered Set	0.2344	0.2060	0.2606	0.0141
	W12	Full Set	0.1965	0.1347	0.2581	0.0257
		Filtered Set	0.4012	0.3646	0.4499	0.0176
Specificity	W0	Full Set	0.9044	0.8960	0.9120	0.0035
		Filtered Set	0.8916	0.8838	0.9006	0.0039
	W4	Full Set	0.9532	0.9310	0.9809	0.0108
		Filtered Set	0.8725	0.8596	0.8862	0.0063
	W8	Full Set	0.9749	0.9665	0.9855	0.0041
		Filtered Set	0.9639	0.9560	0.9701	0.0034
	W12	Full Set	0.9700	0.9499	0.9821	0.0066
		Filtered Set	0.9196	0.9016	0.9352	0.0063

Table 16. Naïve Bayes Models Performance Summary

Week	Features Set	AUC		Sensitivity		Specificity	
		Train	Test	Train	Test	Train	Test
W0	Filtered Set	0.8165	0.8138	0.4742	0.4659	0.8916	0.8930
W4	Filtered Set	0.8205	0.8229	0.5366	0.5411	0.8725	0.8727
W8	Filtered Set	0.8027	0.8050	0.2344	0.2274	0.9639	0.9647
W12	Filtered Set	0.8047	0.8041	0.4012	0.3937	0.9196	0.9196

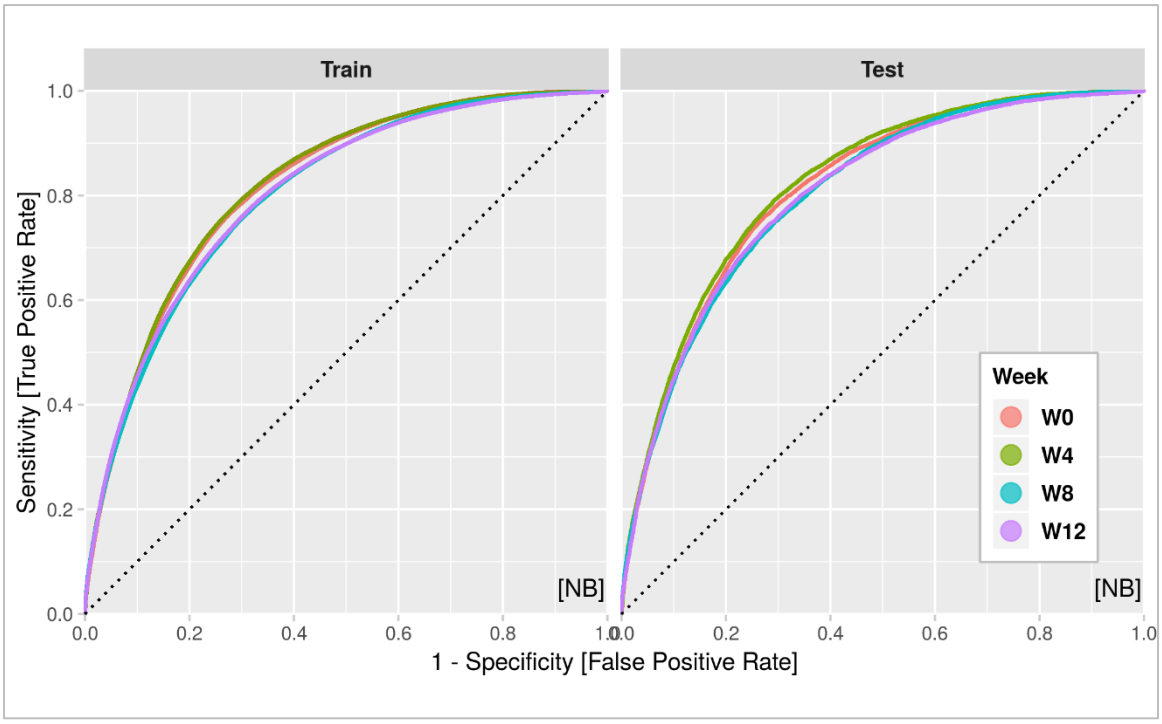

Figure 13. ROC Curves for the NB Models

Table 16 provides a summary of the training and evaluation performance metrics for models trained on the filtered features sets. For week zero dataset, the best model trained on this dataset was able to achieve an AUC of 81%, a sensitivity of 46%, and a specificity of 89%. Week four results attained the best performance across all performance metrics with an AUC of 82%, a sensitivity of 54%, and a specificity of 87%. Overall, it seems that there is no consistent improvement in performance as the semester

progresses towards the end. It is important to note that there is a relationship between the number of features and the performance as a higher number of features resulted in a worse performance. Such a poor performance may be due to the fact that NB models assume all of the input variables are independent of each other.

Artificial Neural Networks

Artificial Neural Networks classifier was trained on two datasets for each of the four experiments. As discussed in the methodology section, ANN algorithm has two tuning parameters: decay and size. Table 17 lists the tuning parameters and the values tried for each one of them. Figure 14 plots the average AUC associated with each of the tuning parameter values. The best tuning parameters used to train each week's final model are listed in Table 18. Figure 14 and Table 18 show that all models trained using 10 hidden units with 0.1 for decay achieved the highest AUC values.

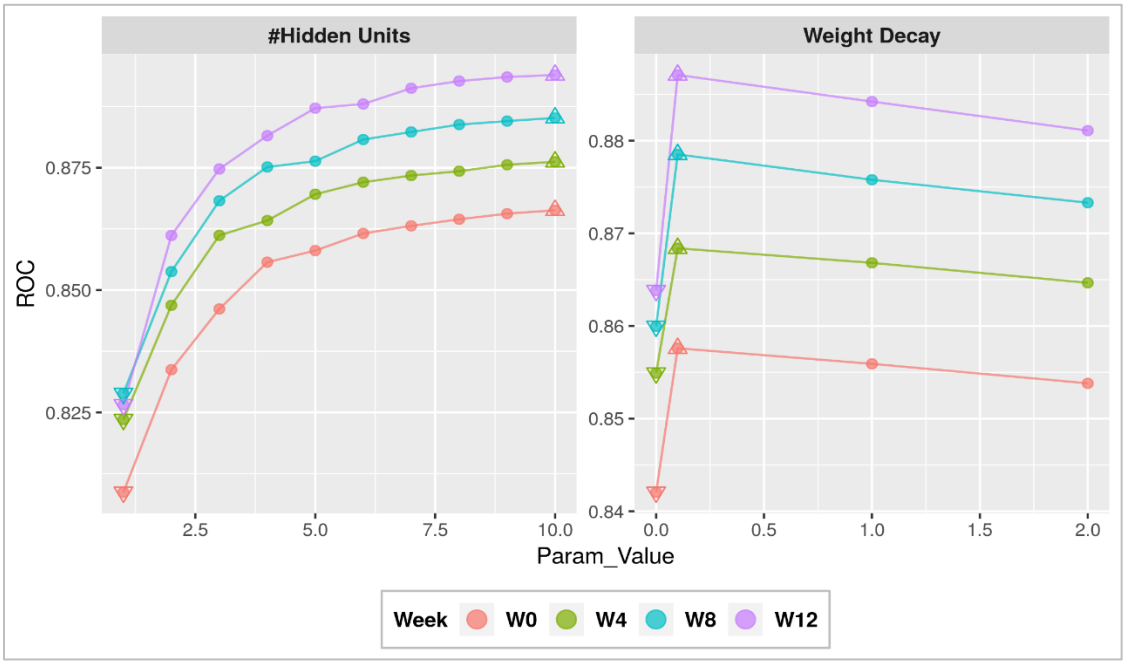

Figure 14. Relationship between ANN Tuning Parameters and AUC

Table 17. List of Artificial Neural Networks Algorithm Tuning Parameters

Parameter	Description	Values	Count
decay	Weight Decay	0, 0.1, 1, 2	4
size	Number of Hidden Units	1, 2, 3, 4, 5, 6, 7, 8, 9, 10	10

Table 18. Artificial Neural Networks Best Tuning Parameters for Each Week's Models

Week	Features Set	Size	Decay
W0	Full Set	10	0.1
W4	Full Set	10	0.1
W8	Full Set	10	0.1
W12	Full Set	10	0.1

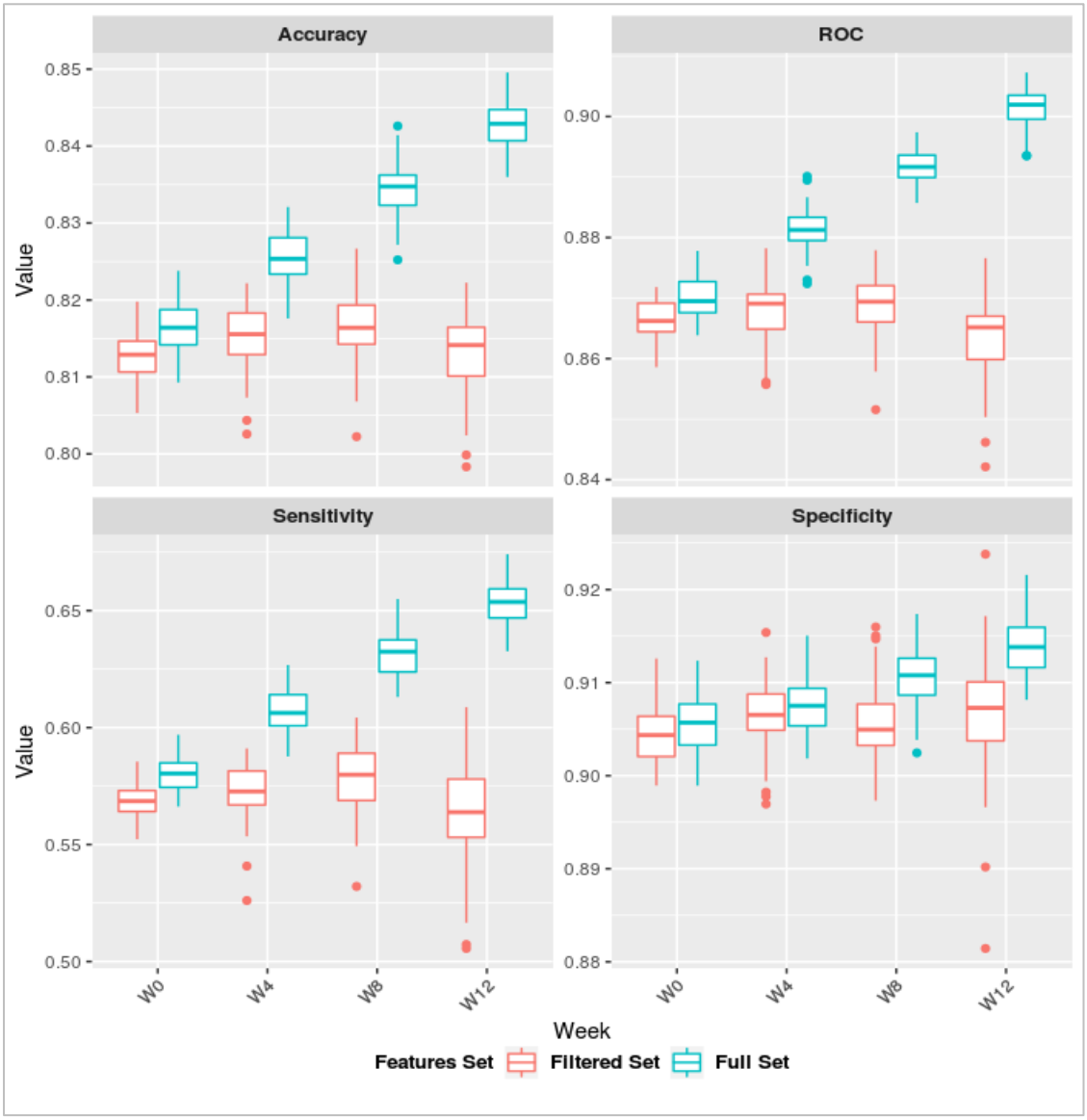

Figure 15. Box-Whisker Plots for the Resampling Distributions of ANN Models

Figure 15 shows a box-whisker plot for the distributions of training performance measures across the 50 models trained for each dataset. The plot shows that the performance of the models trained on the full features sets is better than the models trained on the filtered sets. Additionally, the performance of models trained on the full features sets shows a consistent improvement as the course progresses throughout the semester, while the models trained on the filtered sets suffered a decrease in performance on subsequent weeks. Table 19 provides more details on the distribution of the training performance across all datasets. All the performance metrics across all datasets have a close to zero standard deviation, which indicates that the value of the performance metrics for all of the 50 trained models were close to the reported means. Such consistent performance suggests the suitability of ANN models for the classification of At-Risk students. All further discussion and analysis will only pertain to models trained on the full features sets due to their superior performance.

Table 20 provides a summary of the training and evaluation performance metrics for models trained on the full features' sets. The results of the evaluation performance metrics are almost identical to those obtained from the training stage. Starting with week zero dataset, the best model trained on this dataset was able to achieve an AUC of 87%, a sensitivity of 58%, and a specificity of 90%. Models trained on week four, eight, and 12 datasets sustained a consistent improvement in all performance metrics. As the semester progressed, each week's model was able to improve by 2% in AUC, 2% in sensitivity, and 0.5% in specificity. By week 12, the model was able to achieve about 90% in AUC, 65% in sensitivity, and 92% in specificity. Figure 16 plots the ROC curves for the training and evaluation stage.

Table 19. Artificial Neural Networks Models Training Performance Summary Statistics

Metric	Week	Features Set	Mean	Min	Max	SD
AUC	W0	Full Set	0.8700	0.8638	0.8778	0.0032
		Filtered Set	0.8664	0.8586	0.8718	0.0033
	W4	Full Set	0.8813	0.8724	0.8901	0.0037
		Filtered Set	0.8676	0.8557	0.8782	0.0047
	W8	Full Set	0.8917	0.8857	0.8974	0.0027
		Filtered Set	0.8684	0.8516	0.8779	0.0052
	W12	Full Set	0.9015	0.8935	0.9073	0.0032
		Filtered Set	0.8634	0.8421	0.8766	0.0075
Sensitivity	W0	Full Set	0.5801	0.5662	0.5970	0.0077
		Filtered Set	0.5692	0.5523	0.5855	0.0074
	W4	Full Set	0.6078	0.5877	0.6267	0.0089
		Filtered Set	0.5726	0.5261	0.5911	0.0131
	W8	Full Set	0.6316	0.6131	0.6550	0.0090
		Filtered Set	0.5783	0.5321	0.6043	0.0152
	W12	Full Set	0.6532	0.6326	0.6741	0.0090
		Filtered Set	0.5638	0.5056	0.6087	0.0229
Specificity	W0	Full Set	0.9055	0.8989	0.9124	0.0033
		Filtered Set	0.9044	0.8989	0.9126	0.0031
	W4	Full Set	0.9075	0.9019	0.9151	0.0030
		Filtered Set	0.9064	0.8970	0.9154	0.0037
	W8	Full Set	0.9106	0.9025	0.9174	0.0033
		Filtered Set	0.9059	0.8973	0.9160	0.0042
	W12	Full Set	0.9142	0.9082	0.9216	0.0035
		Filtered Set	0.9068	0.8814	0.9238	0.0066

Table 20. Artificial Neural Networks Models Performance Summary

Week	Features Set	AUC		Sensitivity		Specificity	
		Train	Test	Train	Test	Train	Test
W0	Full Set	0.8700	0.8686	0.5801	0.5832	0.9055	0.9024
W4	Full Set	0.8813	0.8834	0.6078	0.6078	0.9075	0.9076
W8	Full Set	0.8917	0.8928	0.6316	0.6334	0.9106	0.9110
W12	Full Set	0.9015	0.9035	0.6532	0.6543	0.9142	0.9160

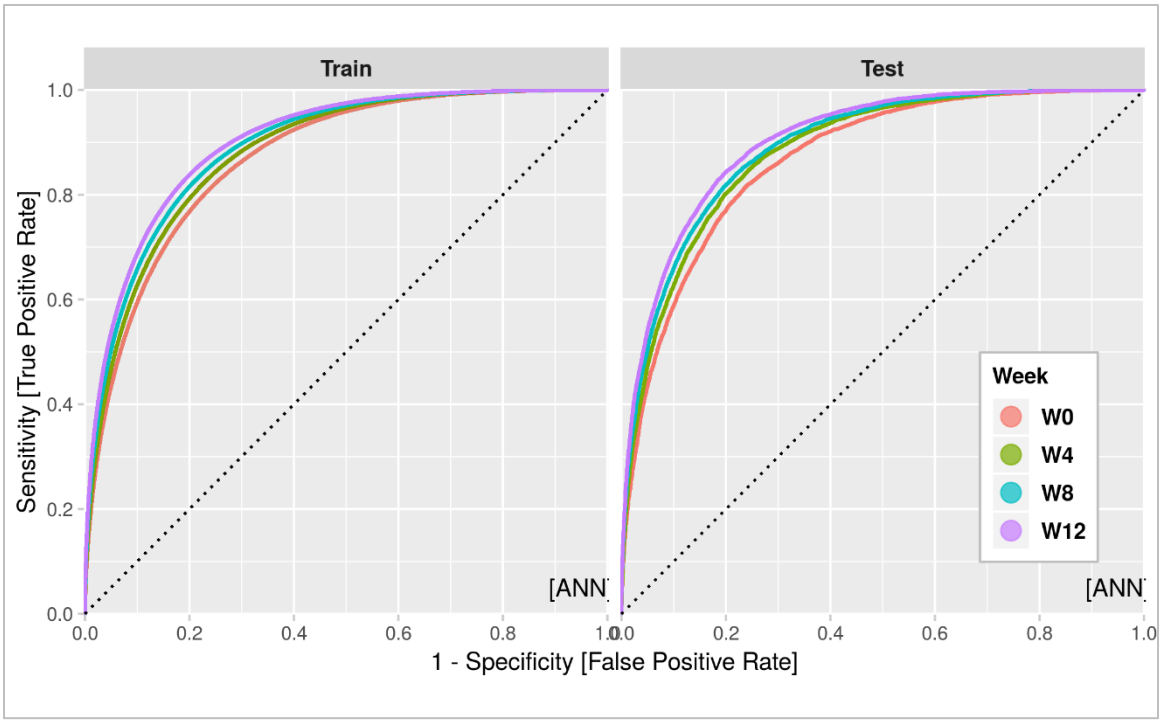

Figure 16. ROC Curves for the ANN Models

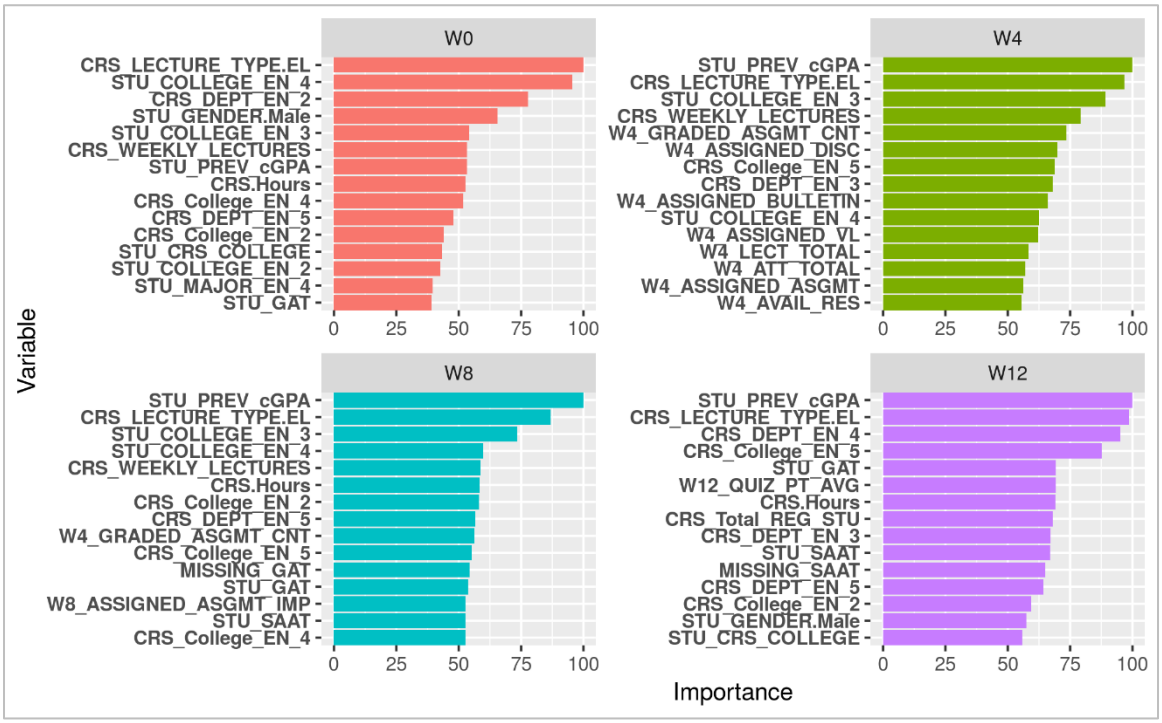

Figure 17. Variable Importance Plots for Each ANN Model

Figure 18 shows the top 40 variables according to their importance to the ANN models, which is determined based on the absolute values of the weights associated with each input variable. The figure shows most of the reported top 40 variables are from the SIS dataset. Figure 17 shows a closer look at the variables' importance for each week's dataset individually. Similar to the overall importance plot, the individual plots show the SIS variables are dominating the top 15 most important variables across all datasets, which may suggest that the predictive power of the SIS variables are much stronger than those in the LMS datasets.

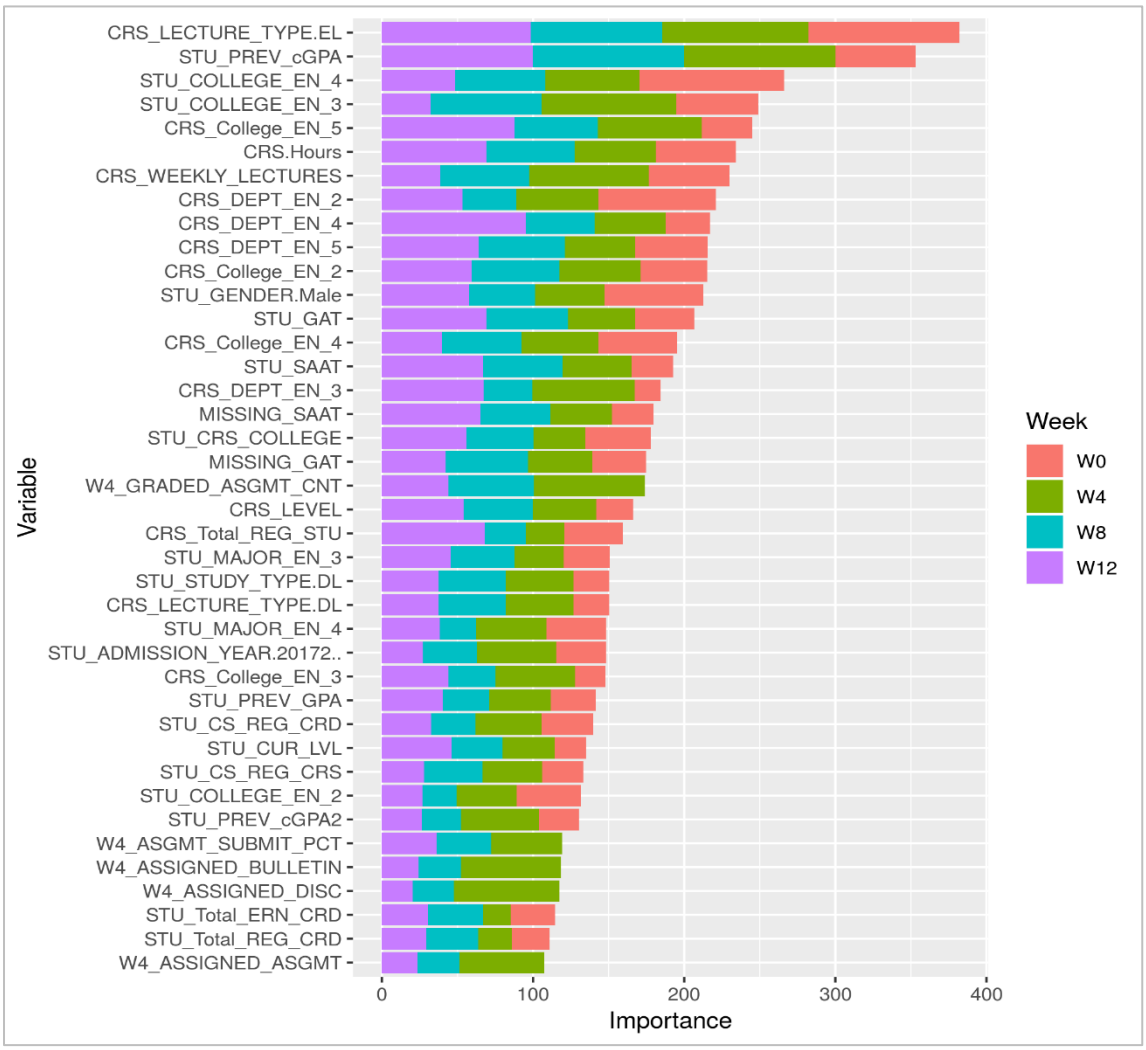

Figure 18. Overall Variables Importance Plot for the ANN Models

Random Forest

Random Forest classifier was trained on two datasets for each of the four experiments. As discussed in the methodology section, RF algorithm has only one tuning parameter: mtry. The mtry tuning parameter refers to the number of predictors randomly selected at each split. For each week's models, 30 different values were tried to fine tune the models. The values ranged from three to the total number of predictors in the dataset. Figure 19 plots the average AUC associated with each value of the mtry tuning parameter. The best tuning parameters used to train each week's final model are listed in Table 21.

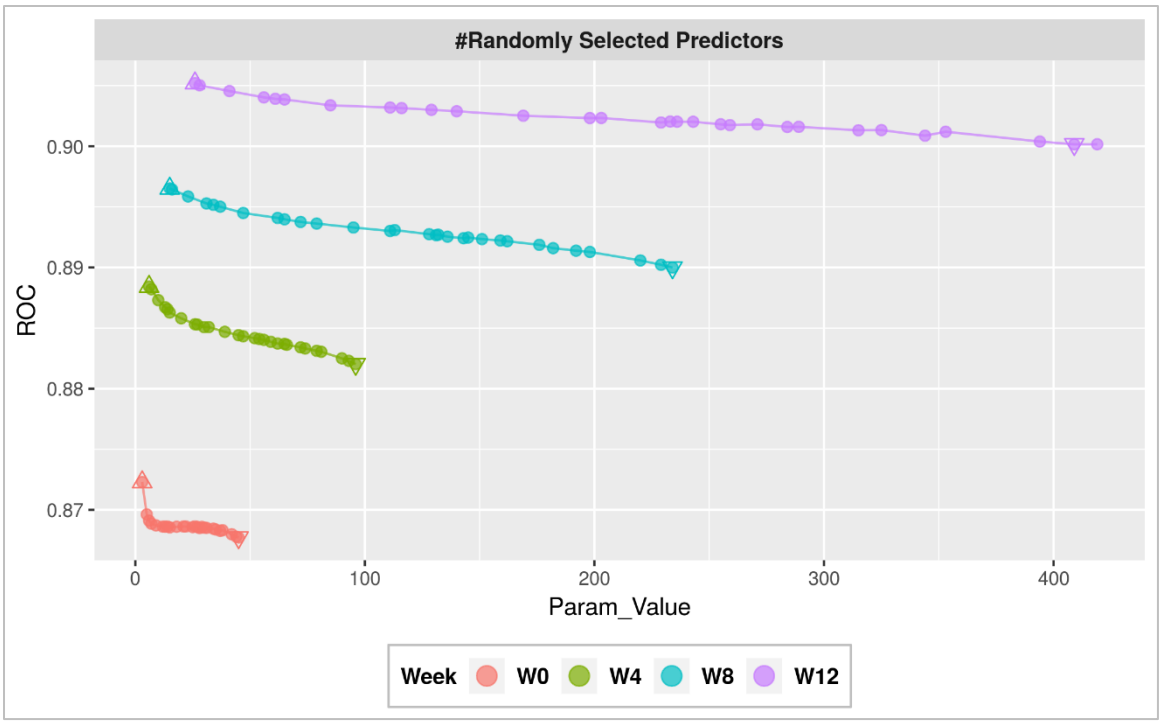

Figure 19. Relationship between RF Tuning Parameter and AUC

Table 21. Random Forest Best Tuning Parameters for Each Week's Models

Week	Features Set	mtry
W0	Full Set	3
W4	Full Set	6
W8	Full Set	15
W12	Full Set	26

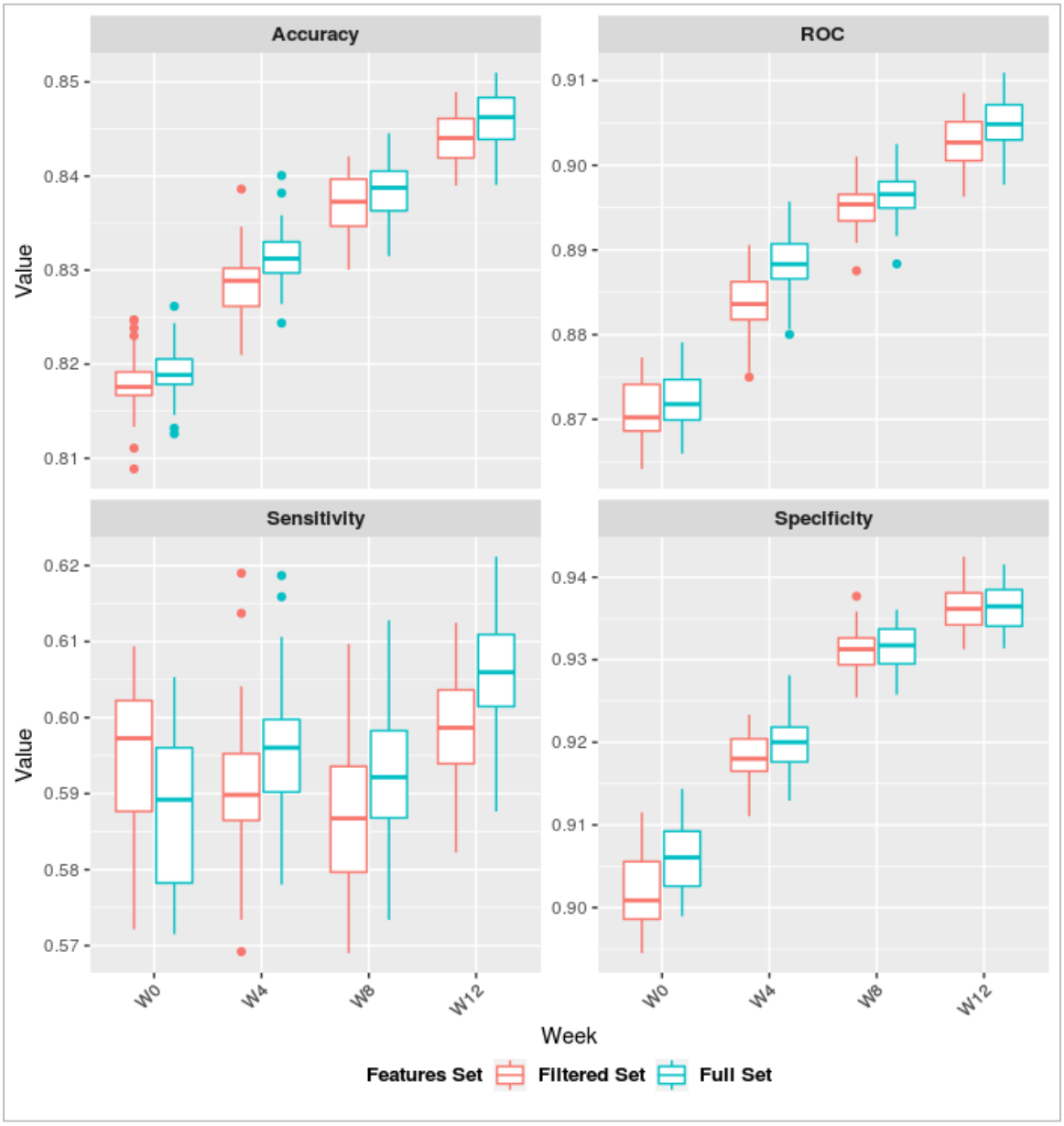

Figure 20. Box-Whisker Plots for the Resampling Distributions of RF Models

Figure 20 shows a box-whisker plot for the distributions of training performance measures across the 50 models trained for each dataset. The plot shows the performance of the models trained on the full features sets is better than the models trained on the filtered sets. While the full features sets' models outperformed the filtered features sets' models, the performance of models trained on both datasets, the full features sets and the filtered features sets, shows a consistent improvement as the course progresses throughout the semester. Table 22 provides more details on the distribution of the training performance across all datasets. Overall, the consistency and stability of RF models across all datasets suggest the suitability of RF models for the classification of At-Risk students. Based on ROC as the criterion measure, all further discussion and analysis will only pertain to models trained on the full features sets since their performance was better than the performance of models trained on the filtered features sets.

Table 23 provides a summary of the training and evaluation performance metrics for models trained on the full features' sets. In general, the results obtained from the training phase were close to those obtained from the evaluation phase. Starting with week zero dataset, the best model trained on this dataset was able to achieve an AUC of 87%, a sensitivity of 59%, and a specificity of 91%. Week four models attained an improved performance reaching 89% on AUC, 60% on sensitivity, and 92% on specificity. Overall, models trained on week four, eight, and 12 datasets sustained a consistent improvement in AUC. By week 12, the model was able to achieve about 90% in AUC, 63% in sensitivity, and 93% in specificity. Figure 21 plots the ROC curves for the training and evaluation stage.

Table 22. Random Forest Models Training Performance Summary Statistics

Metric	Week	Features Set	Mean	Min	Max	SD
AUC	W0	Full Set	0.8723	0.8660	0.8791	0.0032
		Filtered Set	0.8710	0.8642	0.8773	0.0032
	W4	Full Set	0.8884	0.8800	0.8957	0.0033
		Filtered Set	0.8837	0.8750	0.8906	0.0034
	W8	Full Set	0.8965	0.8884	0.9025	0.0026
		Filtered Set	0.8951	0.8875	0.9010	0.0026
	W12	Full Set	0.9050	0.8977	0.9109	0.0029
		Filtered Set	0.9030	0.8963	0.9085	0.0029
Sensitivity	W0	Full Set	0.5881	0.5715	0.6053	0.0101
		Filtered Set	0.5947	0.5721	0.6094	0.0097
	W4	Full Set	0.5959	0.5780	0.6187	0.0080
		Filtered Set	0.5904	0.5692	0.6190	0.0091
	W8	Full Set	0.5922	0.5734	0.6128	0.0096
		Filtered Set	0.5865	0.5690	0.6097	0.0099
	W12	Full Set	0.6059	0.5877	0.6212	0.0079
		Filtered Set	0.5986	0.5822	0.6125	0.0074
Specificity	W0	Full Set	0.9060	0.8989	0.9144	0.0043
		Filtered Set	0.9019	0.8945	0.9115	0.0046
	W4	Full Set	0.9200	0.9129	0.9281	0.0031
		Filtered Set	0.9181	0.9111	0.9233	0.0028
	W8	Full Set	0.9314	0.9258	0.9361	0.0028
		Filtered Set	0.9311	0.9254	0.9377	0.0027
	W12	Full Set	0.9365	0.9314	0.9415	0.0026
		Filtered Set	0.9362	0.9313	0.9425	0.0026

Table 23. Random Forest Networks Models Performance Summary

Week	Features Set	AUC		Sensitivity		Specificity	
		Train	Test	Train	Test	Train	Test
W0	Full Set	0.8723	0.8719	0.5881	0.5858	0.9060	0.9090
W4	Full Set	0.8884	0.8910	0.5959	0.6025	0.9200	0.9216
W8	Full Set	0.8965	0.9002	0.5922	0.5986	0.9314	0.9319
W12	Full Set	0.9050	0.9055	0.6059	0.6132	0.9365	0.9344

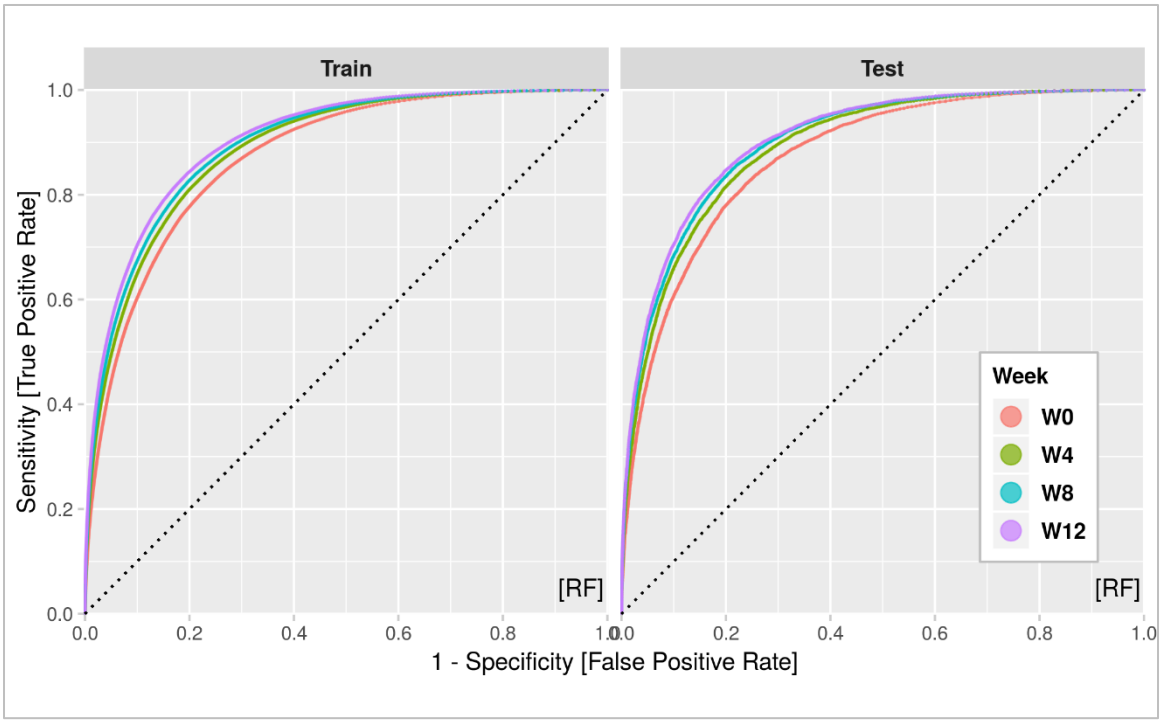

Figure 21. ROC Curves for the RF Models

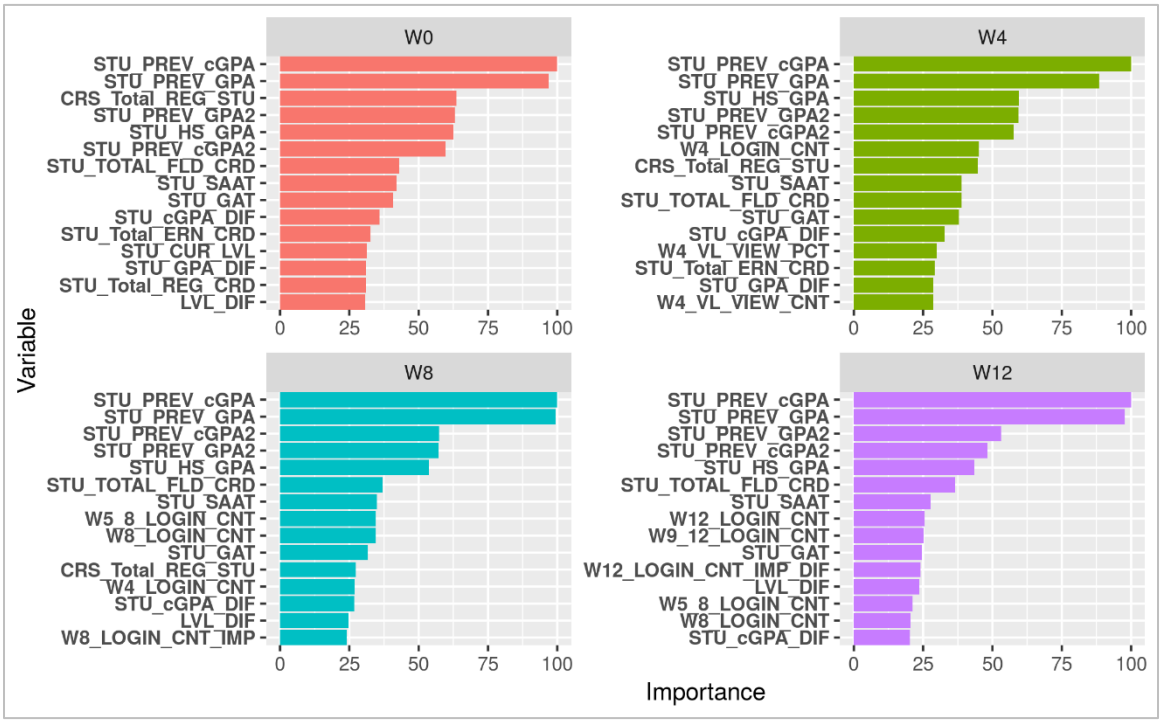

Figure 22. Variable Importance Plots for Each RF Model

Figure 23 shows the top 40 variables according to their importance to the RF models. The figure shows most of the reported top 40 variables are from the SIS dataset. Figure 22 shows a closer look at the variables' importance for each week's dataset individually. Similar to the overall importance plot, the individual plots show the SIS variables are dominating the top 15 most important variables across all datasets, which may suggest that the predictive power of the SIS variables are much stronger than those in the LMS datasets.

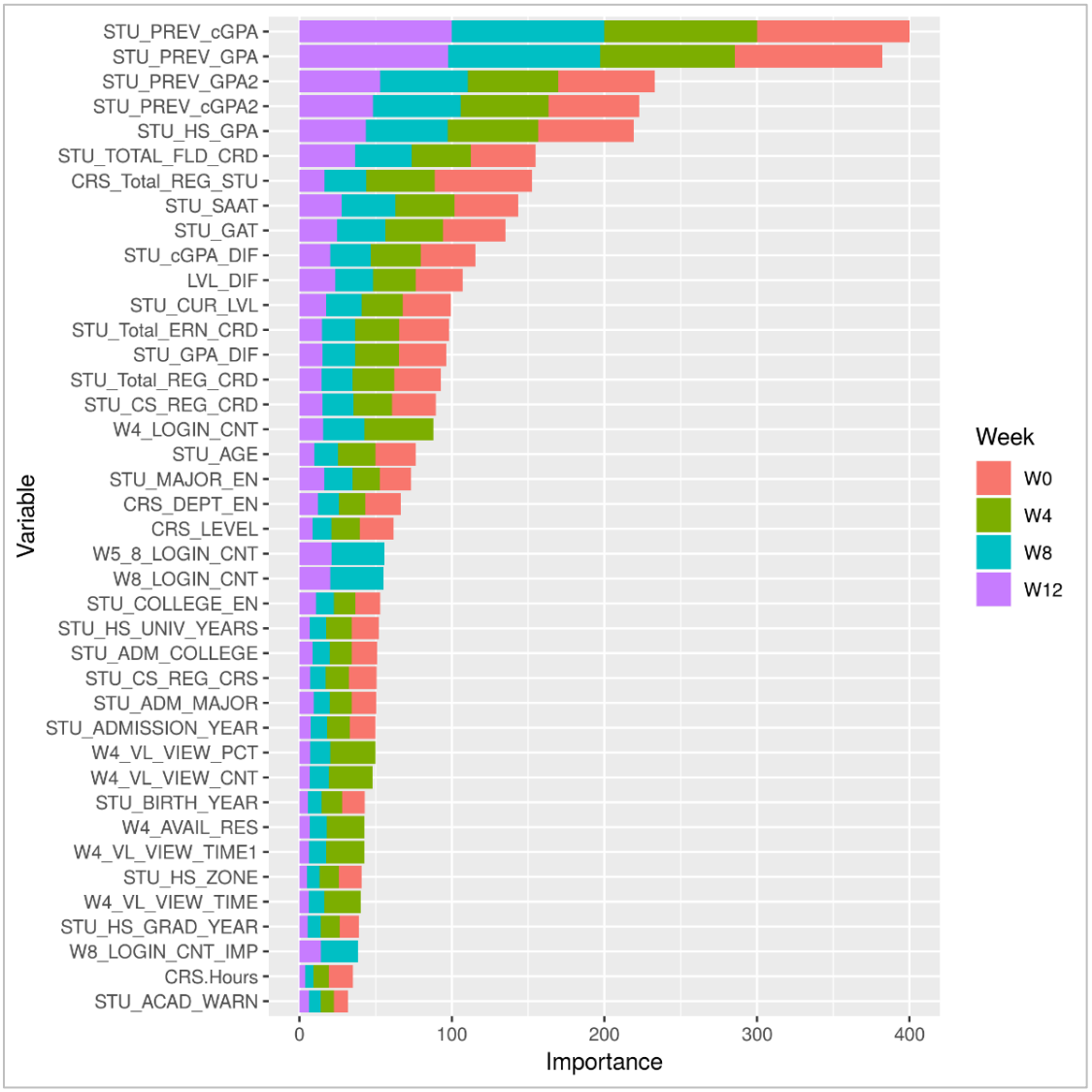

Figure 23. Overall Variables Importance Plot for the RF Models

Extreme Gradient Boosting

Extreme Gradient Boosting classifier was trained on two datasets for each of the four experiments. As discussed in the methodology section, seven parameters of the XGBoost algorithm were fine tuned to optimize the XGBoost models. Table 24 lists the tuning parameters and the values tried for each one of them. Figure 24 plots the average AUC associated with the values of two tuning parameter. The best tuning parameters used to train each week's final model are listed in Tables 25 and 26.

Table 24. List of Extreme Gradient Boosting Algorithm Tuning Parameters

Parameter	Description	Values Range	Count
max_depth	Max Tree Depth	1 – 10	10
min_child_weight	Minimum Sum of Instance Weight	0 – 20	16
nrounds	# Boosting Iterations	61 – 983	30
eta	Shrinkage	0.006 – 0.553	30
gamma	Minimum Loss Reduction	0.766 – 9.968	30
colsample_bytree	Subsample Ratio of Columns	0.334 – 0.695	30
subsample	Subsample Percentage	0.295 – 0.999	30

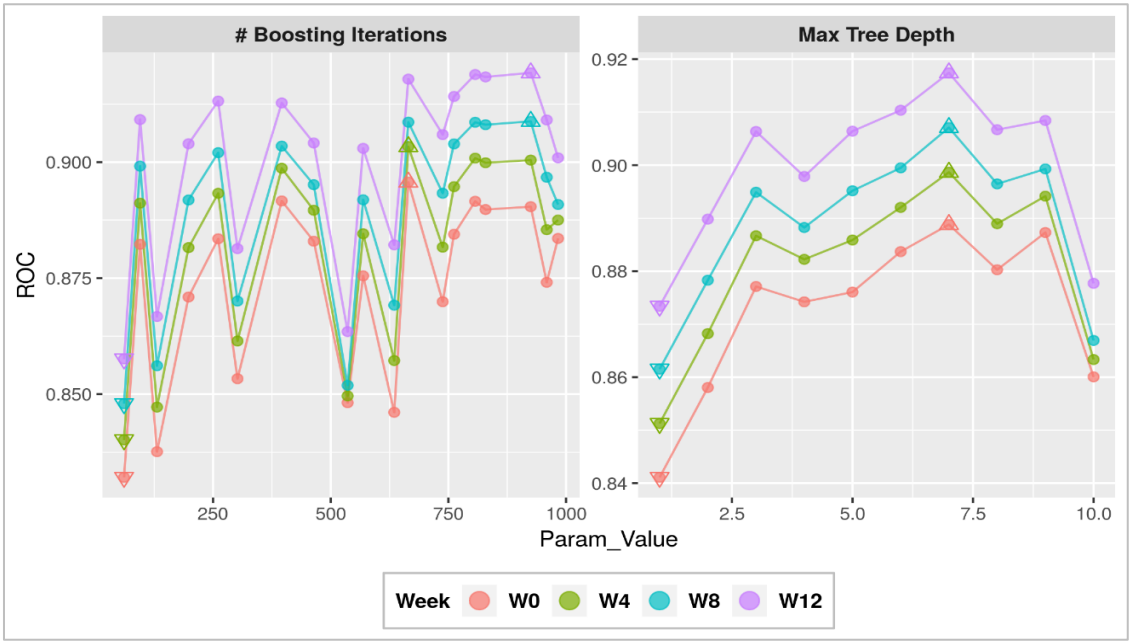

Figure 24. Relationship between XGBoost Tuning Parameters and AUC

Table 25. XGBoost Best Tuning Parameters for Each Week's Models

Week	Features Set	nrounds	max_depth	min_child_weight
W0	Full Set	665	8	10
W4	Full Set	665	8	10
W8	Full Set	925	7	16
W12	Full Set	925	7	16

Table 26. XGBoost Best Tuning Parameters for Each Week's Models

Week	eta	gamma	colsample_bytree	subsample
W0	0.1376	3.5647	0.6301	0.9304
W4	0.1376	3.5647	0.6301	0.9304
W8	0.0591	6.6563	0.5685	0.7931
W12	0.0591	6.6563	0.5685	0.7931

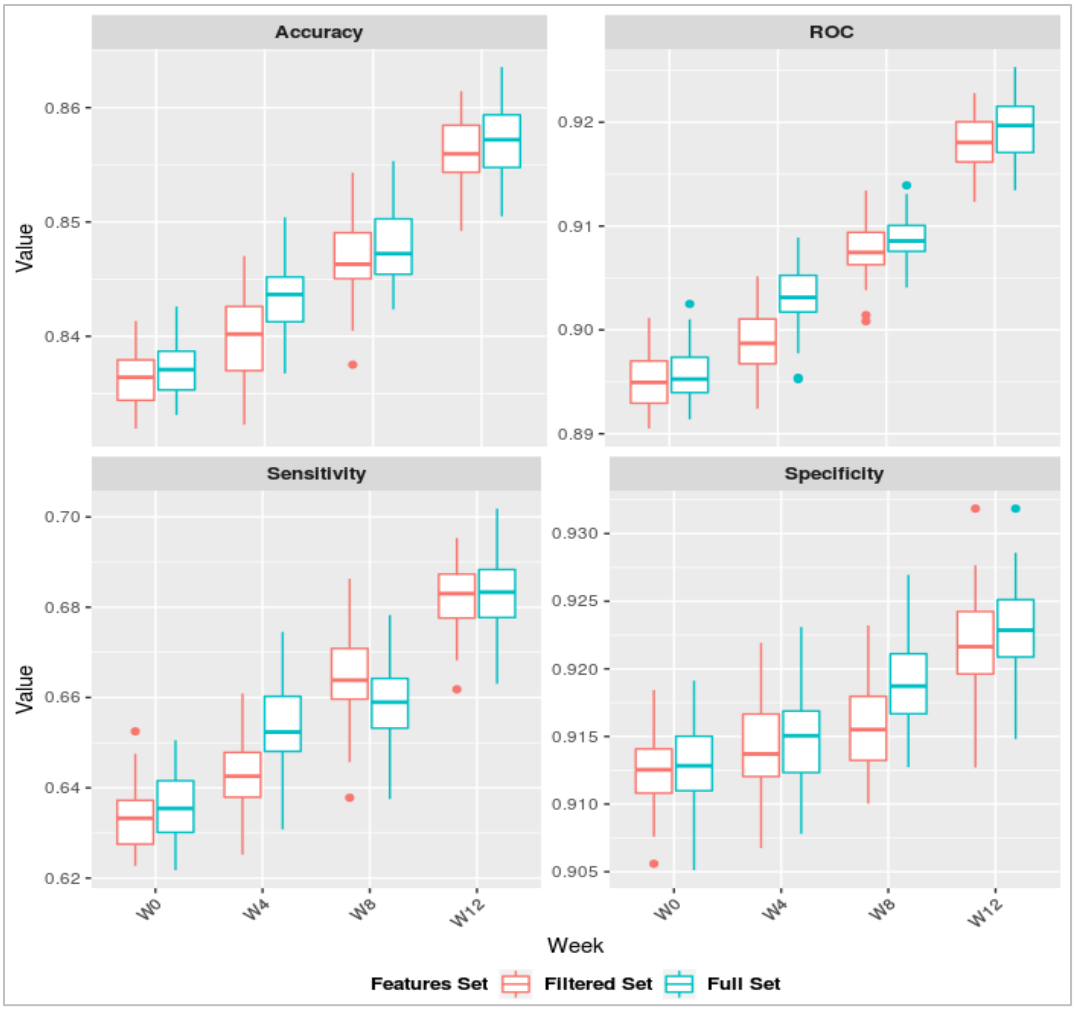

Figure 25. Box-Whisker Plots for the Resampling Distributions of XGBoost Models

Figure 25 shows a box-whisker plot for the distribution of training performance measures across the 50 models trained for each dataset. The plot shows that the performance of the models trained on the full features sets is better than the models trained on the filtered sets. While the full features sets' models outperformed the filtered features sets' models, the performance of models trained on both datasets, the full features sets and the filtered features sets, shows a consistent improvement as the course progresses throughout the semester. Table 27 provides more details on the distribution of the training performance across all datasets. Overall, the consistency and stability of XGBoost models across all datasets suggest the suitability of XGBoost models for the classification of At-Risk students. Based on ROC as the criterion measure, all further discussion and analysis will only pertain to models trained on the full features sets since their performance was better than the performance of models trained on the filtered features sets.

Table 28 provides a summary of the training and evaluation performance metrics for models trained on the full features' sets. In general, the results obtained from the training phase were close to those obtained from the evaluation phase. Starting with week zero dataset, the best model trained on this dataset was able to achieve an AUC of 89%, a sensitivity of 64%, and a specificity of 91%. Week four models attained an improved performance reaching 90% on AUC, 66% on sensitivity, and 92% on specificity. Overall, models trained on week four, eight, and 12 datasets sustained a consistent improvement in all performance metrics. By week 12, the model was able to achieve about 92% in AUC, 69% in sensitivity, and 93% in specificity. Figure 26 plots the ROC curves for the training and evaluation stage.

Table 27. XGBoost Models Training Performance Summary Statistics

Metric	Week	Features Set	Mean	Min	Max	SD
AUC	W0	Full Set	0.8957	0.8914	0.9025	0.0026
		Filtered Set	0.8950	0.8905	0.9012	0.0028
	W4	Full Set	0.9033	0.8953	0.9089	0.0031
		Filtered Set	0.8989	0.8924	0.9052	0.0032
	W8	Full Set	0.9088	0.9041	0.9139	0.0022
		Filtered Set	0.9079	0.9008	0.9134	0.0026
	W12	Full Set	0.9192	0.9134	0.9253	0.0027
		Filtered Set	0.9180	0.9123	0.9228	0.0025
Sensitivity	W0	Full Set	0.6359	0.6218	0.6505	0.0074
		Filtered Set	0.6332	0.6227	0.6525	0.0070
	W4	Full Set	0.6535	0.6308	0.6745	0.0100
		Filtered Set	0.6432	0.6252	0.6609	0.0086
	W8	Full Set	0.6586	0.6375	0.6783	0.0089
		Filtered Set	0.6642	0.6378	0.6863	0.0094
	W12	Full Set	0.6837	0.6630	0.7018	0.0082
		Filtered Set	0.6821	0.6618	0.6953	0.0075
Specificity	W0	Full Set	0.9129	0.9051	0.9191	0.0029
		Filtered Set	0.9127	0.9056	0.9184	0.0028
	W4	Full Set	0.9149	0.9078	0.9231	0.0033
		Filtered Set	0.9141	0.9068	0.9219	0.0036
	W8	Full Set	0.9190	0.9127	0.9269	0.0031
		Filtered Set	0.9156	0.9100	0.9232	0.0032
	W12	Full Set	0.9227	0.9148	0.9318	0.0033
		Filtered Set	0.9218	0.9127	0.9318	0.0033

Table 28. XGBoost Models Performance Summary

Week	Features Set	AUC		Sensitivity		Specificity	
		Train	Test	Train	Test	Train	Test
W0	Full Set	0.8957	0.8958	0.6359	0.6408	0.9129	0.9122
W4	Full Set	0.9033	0.9059	0.6535	0.6586	0.9149	0.9153
W8	Full Set	0.9088	0.9129	0.6586	0.6681	0.9190	0.9207
W12	Full Set	0.9192	0.9192	0.6837	0.6851	0.9227	0.9222

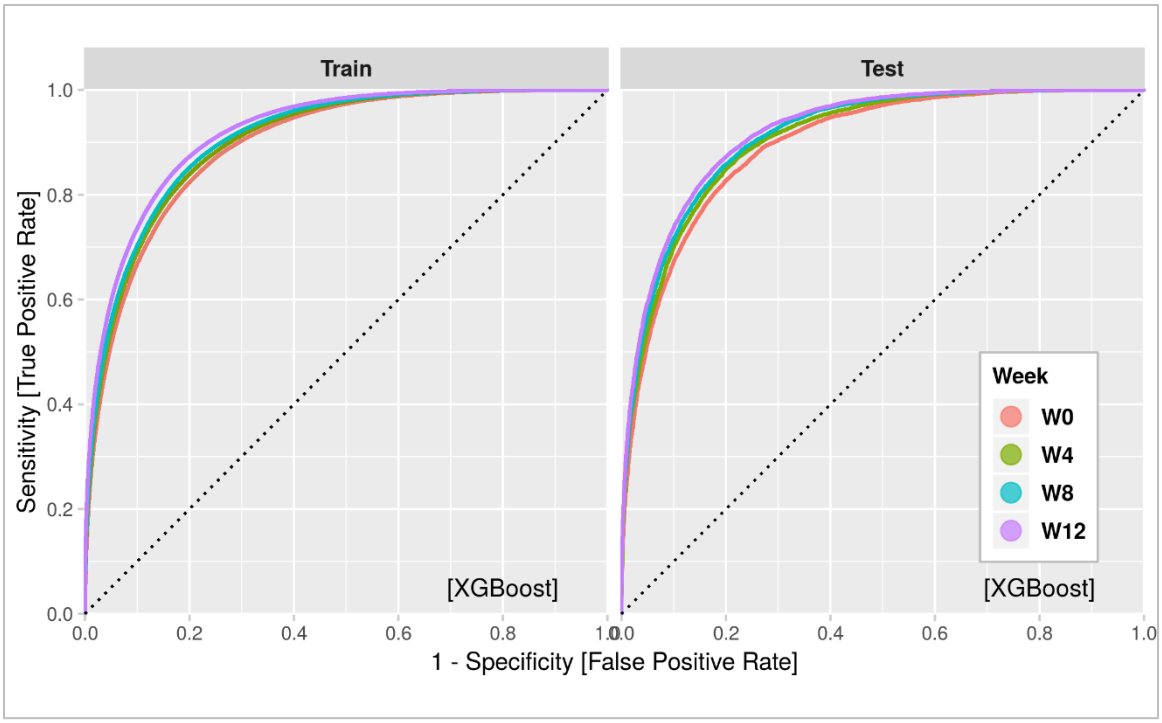

Figure 26. ROC Curves for the XGBoost Models

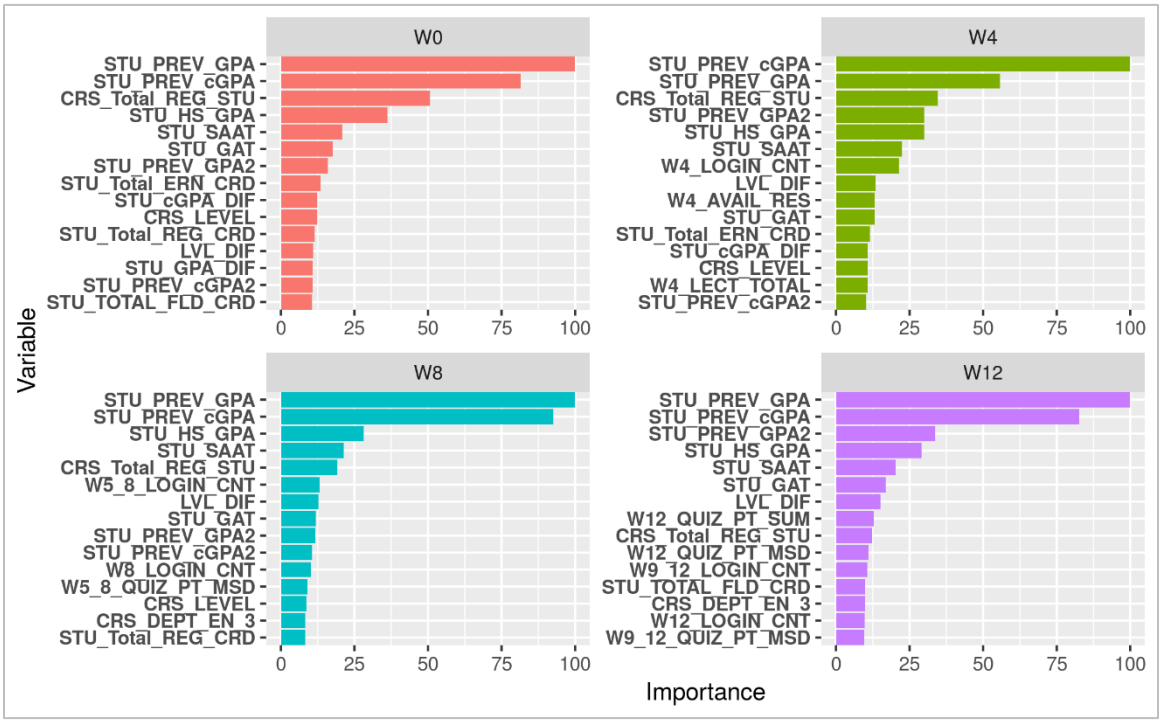

Figure 27. Variable Importance Plots for Each XGBoost Model

Figure 28 shows the top 40 variables according to their importance to the XGBoost models. The figure shows most of the reported top 40 variables are from the SIS dataset. Figure 27 shows a closer look at the variables' importance for each week's dataset individually. Similar to the overall importance plot, the individual plots show the SIS variables are dominating the top 15 most important variables across all datasets, which may suggest that the predictive power of the SIS variables are much stronger than those in the LMS datasets.

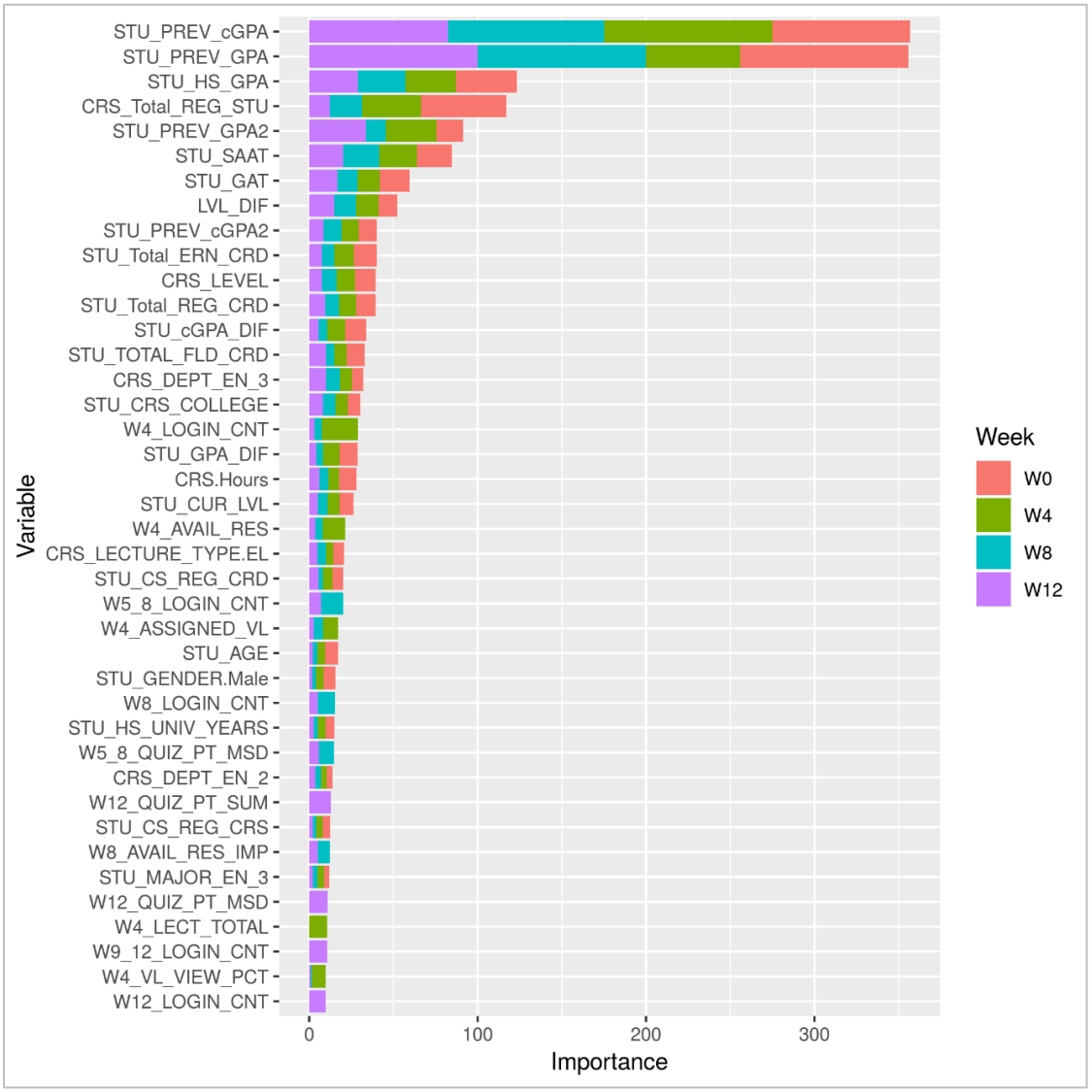

Figure 28. Overall Variables Importance Plot for the XGBoost Models

Comparison of Classifiers

Experiment One (W0 Dataset)

In this experiment, six different classifiers were trained using the same training dataset, week zero's dataset. Week zero's dataset includes data about students and courses that were available before the beginning of the semester; hence, it is called week zero. As shown in Table 8, week zero's full features set included a total of 157,227 observations and 46 variables. Table 29 shows the performance of all classifiers trained on week zero's dataset.

Table 29. Performance of Classifiers Trained on Week Zero's Dataset

Overall Rank	Model	AUC	Sensitivity	Specificity	Type Rank
		Ensemble Classifiers			
1	XGBoost	0.896	0.641	0.912	1
2	RF	0.872	0.586	0.909	2
		Single Classifiers			
3	ANN	0.869	0.583	0.902	1
4	C5	0.833	0.549	0.898	2
5	LR	0.832	0.476	0.906	3
6	NB	0.814	0.466	0.893	4

The table above ranks the classifiers according to their AUC scores. The type rank refers to the rank of classifiers within a group (Single and Ensemble), while the overall rank denotes the classifiers rank without consideration to their type. Starting with the single classifiers, the table shows that the ANN model ranked first with an AUC score of about 87%. The C5 model was found to be the second-best single classifier with an AUC of 83%. As for the ensemble classifiers, the XGBoost model ranked first with an AUC score of about 90%. The RF model scored an AUC of about 87%. The ensemble classifiers were able to improve the performance of single classifiers by at least 1%.

Furthermore, the XGBoost model was able to improve the performance of single classifiers by 3%. Overall, the performance of ensemble classifiers was superior to the performance of single classifiers. As a result, the XGBoost and RF models were ranked first and second. The ANN model was ranked third with an AUC closer to the one obtained by the RF model. The performance of all other single classifiers was at least 4% less than the AUC achieved by the top classifiers. All further analysis and discussion will be based on the XGBoost model since it was the best classifier trained on week zero's dataset.

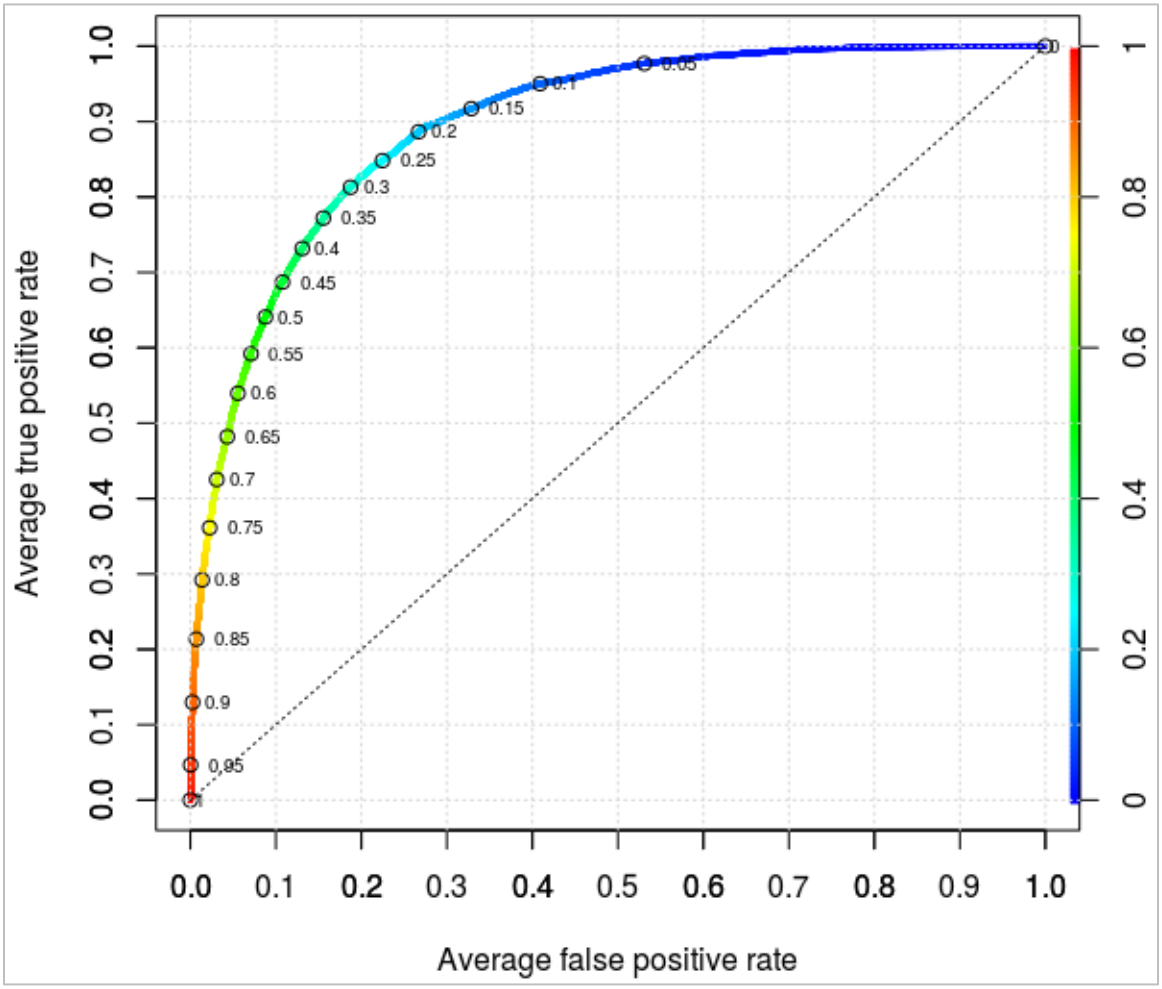

Figure 29. ROC Curve for the XGBoost Trained on Week Zero's Dataset

As previously discussed, the ROC curve plots the resulting sensitivity against the false positive rate for a range of thresholds. The values of sensitivity and specificity reported in the previous section was based on the default cut-off threshold, which is 0.50. This means if the predicted probability of being an at-risk student was above the 0.50 cut-off, the students would be classified as at-risk. Otherwise, the student will be classified as in good-standing. Figure 29 plots the ROC curve for the XGBoost model trained on week zero's dataset along with the threshold values. This curve helps in identifying the cut-off point that maximizes both sensitivity and specificity. The plot shows that the value of the sensitivity increases as the value of the threshold decreases while the value of the specificity decreases as well. Figure 30 plots the values of sensitivity and specificity as a function of threshold. The plot shows that the threshold value that maximizes both the sensitivity and specificity is 0.30, which resulted in sensitivity and specificity of 81%.

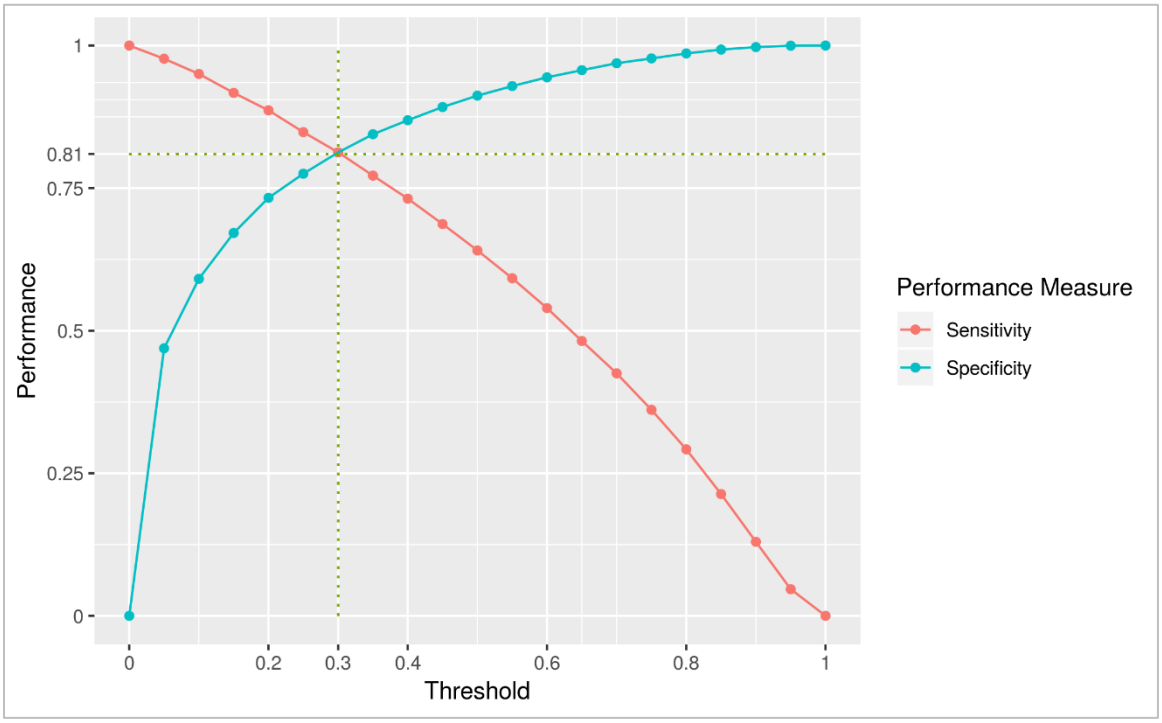

Figure 30. Threshold Analysis for the XGBoost Model Trained on Week Zero's Dataset

To evaluate the generalizability of week zero's XGBoost model across on-campus and online students, the original test set previously used to evaluate the XGBoost was split into two test sets based on students' types: Campus Learning (CL) students and Distance Learning (DL) students. The CL test set included the test set data concerning on-campus students only, while the DL test set included the test set data concerning online students only. Additionally, week zero's training set was split into two training sets using the same procedure followed for splitting the test set. Next, using the two training sets, two more XGBoost models were trained using the best tuning parameters identified in the previous section. As a result, two models were trained and evaluated, one trained on on-campus students' data only (will be referred to as CL Students Model) and the other trained on online students' data only (will be referred to as DL Students Model). Finally, the original model trained previously will be referred to as ALL Students Model.

Table 30. Results of the Generalizability Analysis for Week Zero's XGBoost Model

Test Set	ALL Students Model			CL Students Model			DL Students Model		
	AUC	**Sen**	**Spec**	**AUC**	**Sen**	**Spec**	**AUC**	**Sen**	**Spec**
ALL Students	**0.896**	**0.813**	**0.813**	0.844	0.780	0.754	0.827	0.624	0.835
CL Students	0.893	0.799	0.820	**0.893**	**0.786**	**0.824**	0.774	0.461	0.853
DL Students	0.899	0.831	0.801	0.769	0.773	0.641	**0.902**	**0.835**	**0.806**

Table 30 presents the results of the generalizability analysis performed on week zero's best model. The sensitivity and specificity reported in the table are based on the 0.30 cut-off threshold identified previously. Starting with ALL Students Model, this model was trained using data that included both on-campus and online students. Then,

the model was evaluated using three test sets: ALL Students, CL Students, and DL Students test sets. Based on ALL Students test set, the model was able to achieve an AUC of 89%, a sensitivity of 81%, and a specificity of 81%. Additionally, the model was able to achieve a similar AUC score using the other test sets. Such consistent performance indicates that the original models trained on all-students data generalizes well across on-campus and online student populations. The results of the CL Students model show a similar AUC score when evaluated using the CL Students test set. However, when evaluated using a different student type test set, such as the DL Students test set, the model performed poorly with an AUC of about 77%. Finally, the results of the DL Students model were similar to the original model when evaluated using the DL Students test set but resulted in a poor performance when evaluated using the CL Students test set.

Experiment Two (W4 Dataset)

In this experiment, six different classifiers were trained using the same training dataset, week four's dataset. Week four's dataset comprised of week zero's dataset in addition to the available data by the end of week four. As shown in Table 8, week four's full features set included a total of 157,227 observations and 98 variables.

Table 31. Performance of Classifiers Trained on Week Four's Dataset

Overall Rank	Model	AUC	Sensitivity	Specificity	Type Rank
		Ensemble Classifiers			
1	XGBoost	0.906	0.659	0.915	1
2	RF	0.891	0.602	0.922	2
		Single Classifiers			
3	ANN	0.883	0.608	0.908	1
4	LR	0.849	0.509	0.908	2
5	C5	0.839	0.566	0.899	3
6	NB	0.823	0.541	0.873	4

Table 31 shows the performance of all classifiers trained on week four's dataset. Starting with the single classifiers, the table shows that the ANN model ranked first with an AUC score of about 88%. The LR model was found to be the second-best single classifier with an AUC of 85%. As for the ensemble classifiers, the XGBoost model ranked first with an AUC score of about 91%. The RF model scored an AUC of 89%. The ensemble classifiers were able to improve the performance of single classifiers by at least 1%. Furthermore, the XGBoost model was able to improve the performance of single classifiers by 2%. Overall, the performance of ensemble classifiers was superior to the performance of single classifiers. As a result, the XGBoost and RF models were ranked first and second. The ANN model was ranked third with an AUC closer to the one obtained by the RF model. The performance of all other single classifiers was at least 3% less than the AUC achieved by the top classifiers. All further analysis and discussion will be based on the XGBoost model since it was the best classifier trained on week four's dataset.

Figure 31 plots the ROC curve for the XGBoost model trained on week four's dataset along with the threshold values. This curve helps in identifying the cut-off point that maximizes both sensitivity and specificity. The plot shows that the value of the sensitivity increases as the value of the threshold decreases while the value of the specificity decreases as well. To demonstrate the relationship between the threshold values and the values of the classification performance measures, Figure 32 plots the values of sensitivity and specificity as a function of threshold. Similar to week zero's model, the plot shows that the threshold value that maximizes both the sensitivity and specificity is 0.30, which resulted in sensitivity and specificity of 82%.

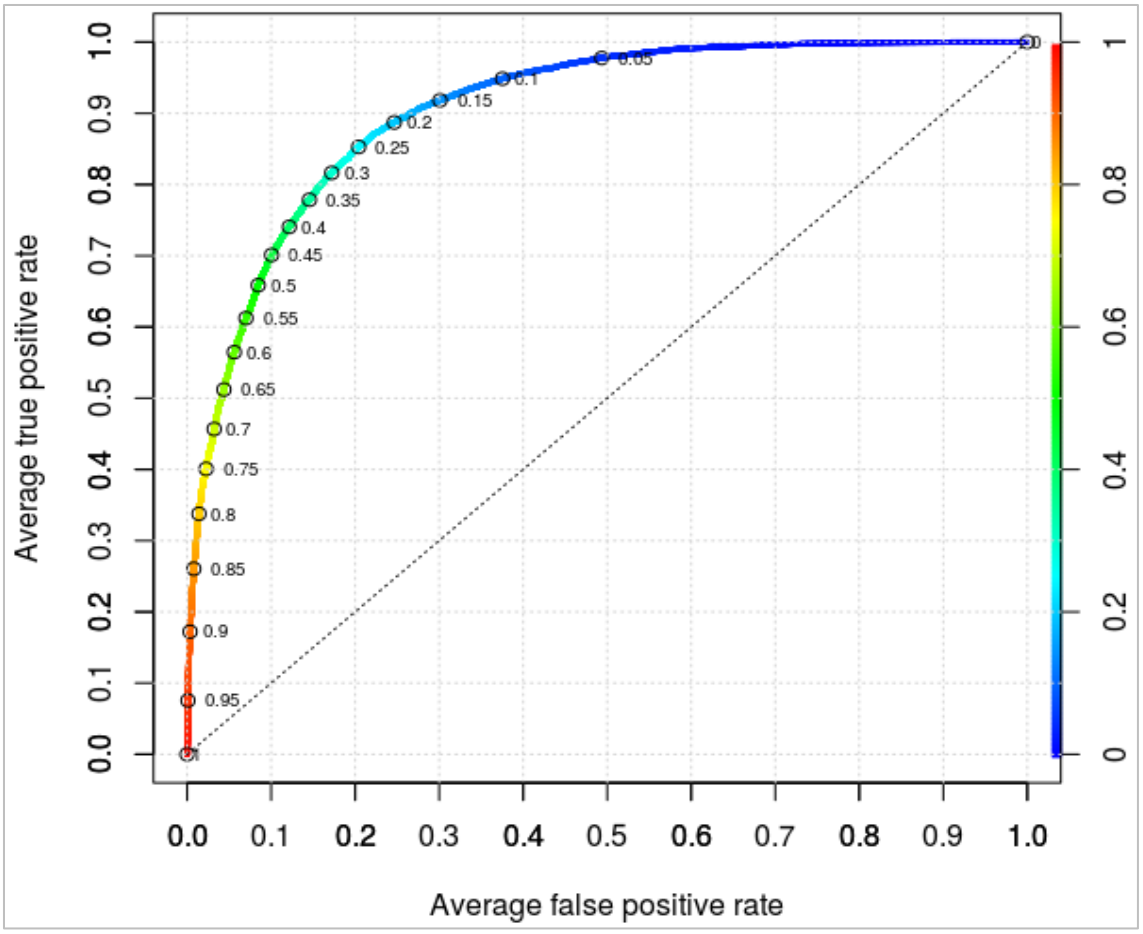

Figure 31. ROC Curve for the XGBoost Trained on Week Four's Dataset

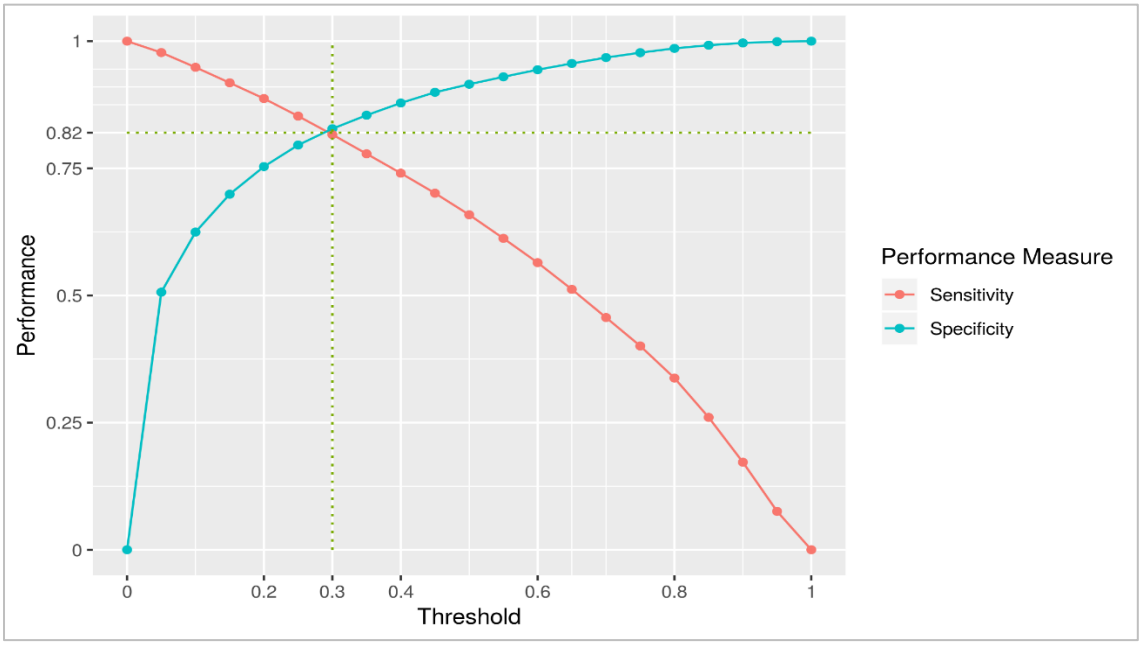

Figure 32. Threshold Analysis for the XGBoost Model Trained on Week Four's Dataset

Table 32 presents the results of the generalizability analysis performed on week four's best model. The sensitivity and specificity reported in the table are based on the 0.30 cut-off threshold identified previously. Based on ALL Students test set, ALL Students Model was able to achieve an AUC of 90%, a sensitivity of 82%, and a specificity of 82%. Additionally, the model was able to achieve a similar AUC score using the other test sets. Such consistent performance indicates that the original model trained on all-students data generalizes well across on-campus and online student populations. The results of the CL Students model show a similar AUC score when evaluated using the CL Students test set. However, when evaluated using a different student type test set, such as the DL Students test set, the model performed poorly with an AUC of about 81%. Finally, the results of the DL Students model, when evaluated using the DL Students test set, were similar to the results of the original model. However, it performed poorly when evaluated using the CL Students test set.

Table 32. Results of the Generalizability Analysis for Week Four's XGBoost Model

Test Set	ALL Students Model			CL Students Model			DL Students Model		
	AUC	**Sen**	**Spec**	**AUC**	**Sen**	**Spec**	**AUC**	**Sen**	**Spec**
ALL Students	**0.906**	**0.816**	**0.828**	0.863	0.733	0.823	0.842	0.670	0.829
CL Students	0.905	0.800	0.840	**0.905**	**0.795**	**0.841**	0.793	0.542	0.837
DL Students	0.906	0.838	0.808	0.809	0.653	0.795	**0.907**	**0.836**	**0.818**

Experiment Three (W8 Dataset)

In this experiment, six different classifiers were trained using the same training dataset, week eight's dataset. Week eight's dataset comprised of week four's dataset in

addition to the available data by the end of week eight. As shown in Table 8, week eight's full features set included a total of 157,227 observations and 239 variables.

Table 33. Performance of Classifiers Trained on Week Eight's Dataset

Overall Rank	Model	AUC	Sensitivity	Specificity	Type Rank
		Ensemble Classifiers			
1	XGBoost	0.913	0.668	0.921	1
2	RF	0.900	0.599	0.932	2
		Single Classifiers			
3	ANN	0.893	0.633	0.911	1
4	LR	0.859	0.541	0.911	2
5	C5	0.843	0.603	0.888	3
6	NB	0.805	0.227	0.965	4

Table 33 shows the performance of all classifiers trained on week eight's dataset. Starting with the single classifiers, the table shows that the ANN model ranked first with an AUC score of 89%. The LR model was found to be the second-best single classifier with an AUC of 86%. As for the ensemble classifiers, the XGBoost model ranked first with an AUC score of about 91%. The RF model scored an AUC of 90%. The ensemble classifiers were able to improve the performance of single classifiers by at least 1%. Furthermore, the XGBoost model was able to improve the performance of single classifiers by 2%. Overall, the performance of ensemble classifiers was superior to the performance of single classifiers. As a result, the XGBoost and RF models were ranked first and second. The ANN model was ranked third with an AUC closer to the one obtained by the RF model. The performance of all other single classifiers was at least 3% less than the AUC achieved by the top classifiers. All further analysis and discussion will be based on the XGBoost model since it was the best classifier trained on week four's dataset.

Figure 33 plots the ROC curve for the XGBoost model trained on week eight's dataset along with the threshold values. This curve helps in identifying the cut-off point that maximizes both sensitivity and specificity. The plot shows that the value of the sensitivity increases as the value of the threshold decreases while the value of the specificity decreases as well. To demonstrate the relationship between the threshold values and the values of the classification performance measures, Figure 34 plots the values of sensitivity and specificity as a function of threshold. Similar to the previous experiments, the plot shows that the threshold value that maximizes both the sensitivity and specificity is 0.30, which resulted in sensitivity and specificity of 83%.

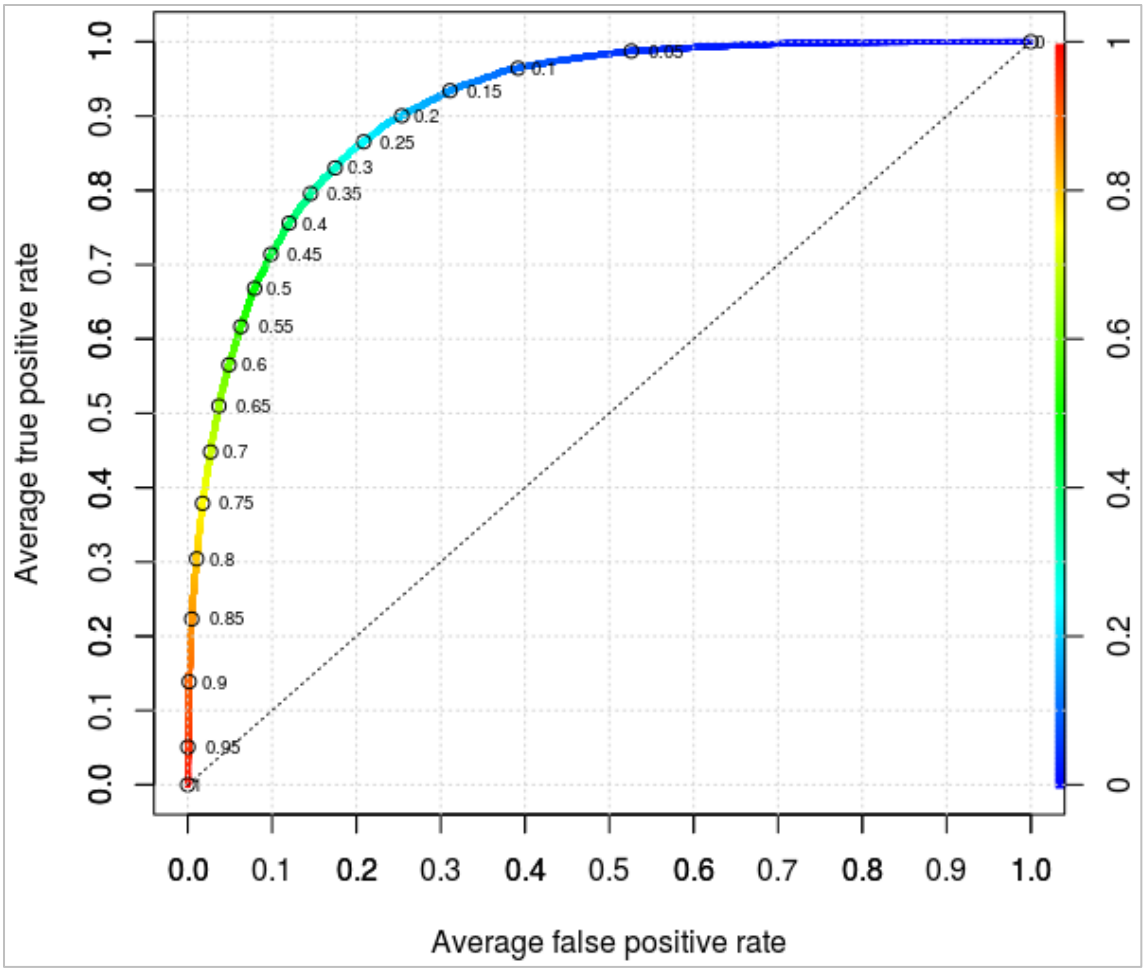

Figure 33. ROC Curve for the XGBoost Trained on Week Eight's Dataset

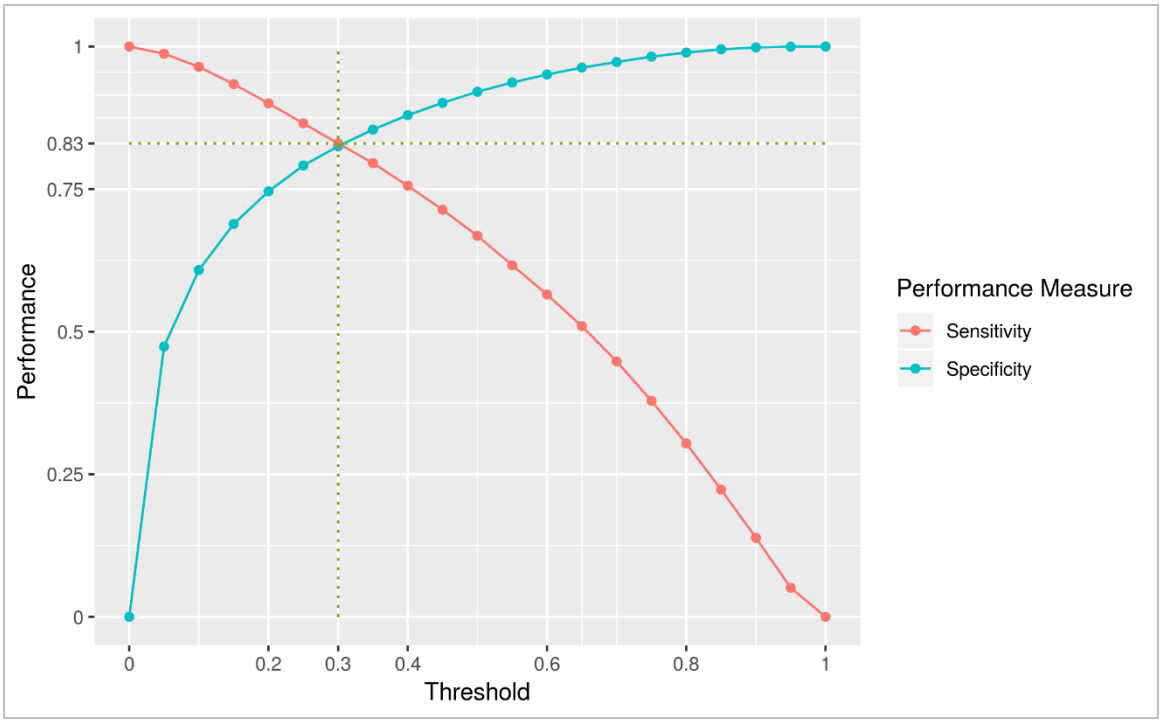

Figure 34. Threshold Analysis for the XGBoost Model Trained on Week Eight's Dataset

Table 34 presents the results of the generalizability analysis performed on week eight's best model. The sensitivity and specificity reported in the table are based on the 0.30 cut-off threshold identified previously. Based on ALL Students test set, ALL Students Model was able to achieve an AUC of 91%, a sensitivity of 83%, and a specificity of 83%. Additionally, the model was able to achieve a similar AUC score using the other test sets. Such consistent performance indicates that the original model trained on all-students data generalizes well across on-campus and online student populations. The results of the CL Students model show a similar AUC score when evaluated using the CL Students test set. However, when evaluated using a different student type test set, such as the DL Students test set, the model performed poorly with an AUC of about 77%. Finally, the results of the DL Students model, when evaluated using

the DL Students test set, were similar to the results of the original model. Nevertheless, it performed poorly when evaluated using the CL Students test set.

Table 34. Results of the Generalizability Analysis for Week Eight's XGBoost Model

Test Set	ALL Students Model			CL Students Model			DL Students Model		
	AUC	**Sen**	**Spec**	**AUC**	**Sen**	**Spec**	**AUC**	**Sen**	**Spec**
ALL Students	**0.913**	**0.830**	**0.825**	0.874	0.754	0.819	0.842	0.729	0.783
CL Students	0.914	0.810	0.841	**0.915**	**0.809**	**0.846**	0.782	0.635	0.764
DL Students	0.910	0.857	0.800	0.810	0.683	0.774	**0.914**	**0.851**	**0.813**

Experiment Four (W12 Dataset)

In this experiment, six different classifiers were trained using the same training dataset, week 12's dataset. Week 12's dataset comprised of week eight's dataset in addition to the available data by the end of week 12. As shown in Table 8, week 12's full features set included a total of 157,227 observations and 427 variables.

Table 35. Performance of Classifiers Trained on Week 12's Dataset

Overall Rank	Model	AUC	Sensitivity	Specificity	Type Rank
		Ensemble Classifiers			
1	XGBoost	0.919	0.685	0.922	1
2	RF	0.906	0.613	0.934	2
		Single Classifiers			
3	ANN	0.904	0.654	0.916	1
4	LR	0.868	0.569	0.911	2
5	C5	0.845	0.580	0.899	3
6	NB	0.804	0.394	0.920	4

Table 35 shows the performance of all classifiers trained on week 12's dataset. Starting with the single classifiers, the table shows that the ANN model ranked first with an AUC score of 90%. The LR model was found to be the second-best single classifier

with an AUC of 87%. As for the ensemble classifiers, the XGBoost model ranked first with an AUC score of about 92%. The RF model scored an AUC of 91%. The ensemble classifiers were able to improve the performance of single classifiers by at least 1%. Furthermore, the XGBoost model was able to improve the performance of single classifiers by 2%. Overall, the performance of ensemble classifiers was superior to the performance of single classifiers. As a result, the XGBoost and RF models were ranked first and second. The ANN model was ranked third with an AUC closer to the one obtained by the RF model. The performance of all other single classifiers was at least 3% less than the AUC achieved by the top classifiers. All further analysis and discussion will be based on the XGBoost model since it was the best classifier trained on week four's dataset.

Figure 35 plots the ROC curve for the XGBoost model trained on week 12's dataset along with the threshold values. This curve helps in identifying the cut-off point that maximizes both sensitivity and specificity. The plot shows that the value of the sensitivity increases as the value of the threshold decreases while the value of the specificity decreases as well. To demonstrate the relationship between the threshold values and the values of the classification performance measures, Figure 36 plots the values of sensitivity and specificity as a function of threshold. Similar to the previous experiments, the plot shows that the threshold value that maximizes both the sensitivity and specificity is 0.30, which resulted in sensitivity and specificity of 84%.

Table 36 presents the results of the generalizability analysis performed on week 12's best model. The sensitivity and specificity reported in the table are based on the 0.30 cut-off threshold identified previously. Based on ALL Students test set, ALL

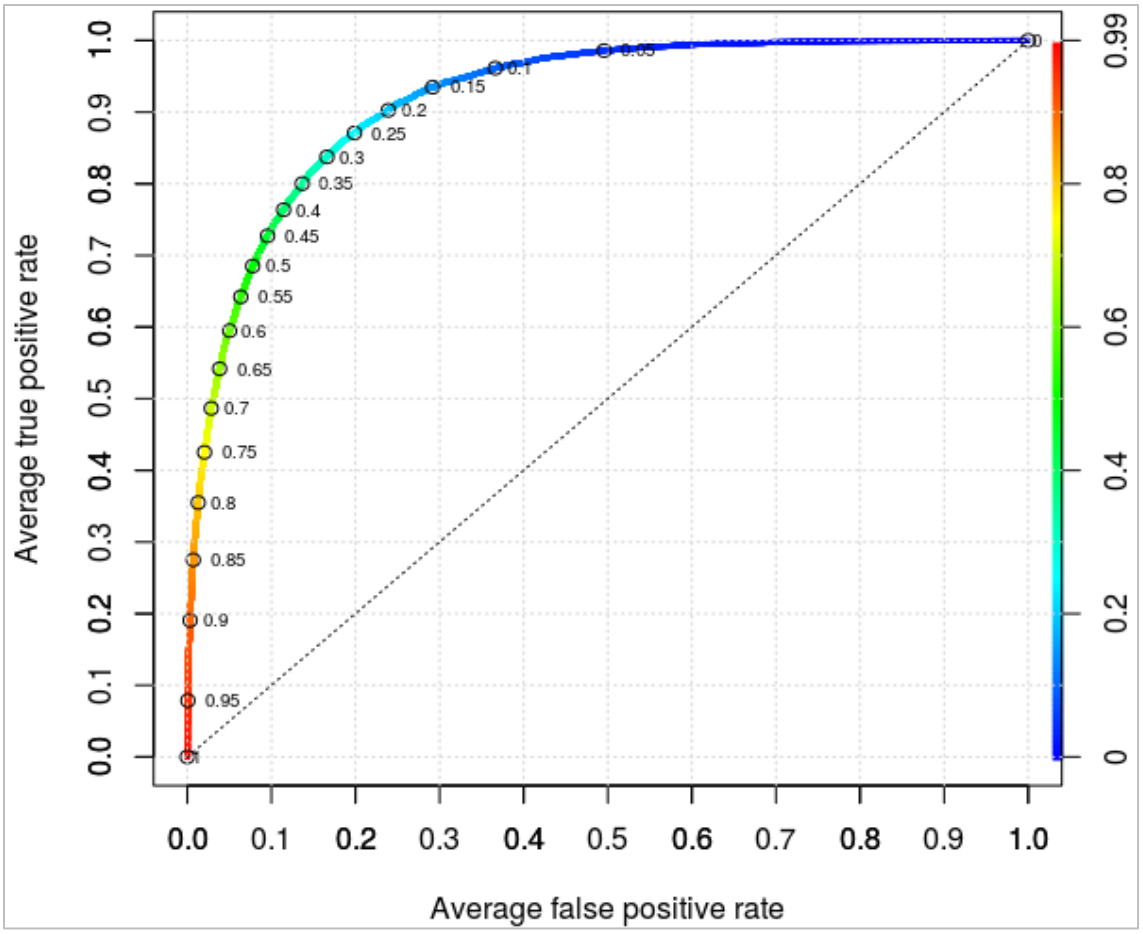

Figure 35. ROC Curve for the XGBoost Trained on Week 12's Dataset

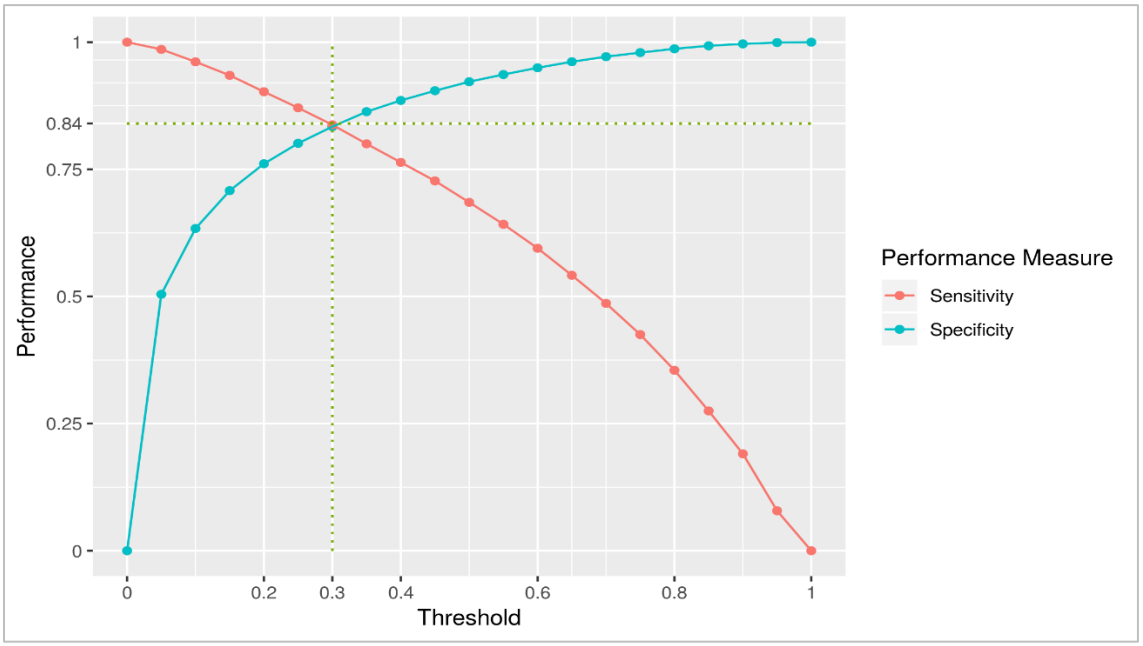

Figure 36. Threshold Analysis for the XGBoost Model Trained on Week 12s Dataset

Students Model was able to achieve an AUC of 92%, a sensitivity of 84%, and a specificity of 83%. Additionally, the model was able to achieve a similar AUC score using the other test sets. Such consistent performance indicates that the original model trained on all-students data generalizes well across on-campus and online student populations. The results of the CL Students model show a similar AUC score when evaluated using the CL Students test set. However, when evaluated using a different student type test set such as, the DL Students test set, the model performed poorly with an AUC of about 82%. Finally, the results of the DL Students model, when evaluated using the DL Students test set, were similar to the results of the original model. Nevertheless, it performed poorly when evaluated using the CL Students test set.

Table 36. Results of the Generalizability Analysis for Week 12's XGBoost Model

Test Set	ALL Students Model			CL Students Model			DL Students Model		
	AUC	**Sen**	**Spec**	**AUC**	**Sen**	**Spec**	**AUC**	**Sen**	**Spec**
ALL Students	**0.919**	**0.837**	**0.834**	0.874	0.694	0.859	0.861	0.794	0.759
CL Students	0.922	0.826	0.847	**0.921**	**0.817**	**0.853**	0.814	0.750	0.722
DL Students	0.915	0.852	0.812	0.816	0.535	0.870	**0.918**	**0.851**	**0.820**

Chapter 5

Conclusions, Implications, Recommendations, and Summary

Conclusion

This dissertation focused on the evaluation of machine learning techniques for early identification of at-risk students. The results of the experiments conducted as part of this research have shown that ML models can identify at-risk students with satisfactory accuracy rates even before the beginning of the classes. To guide the evaluation process, six research questions were proposed and answered through the analysis of the experiments results. The research questions and their findings are listed below:

1) What are the best ML techniques to predict at-risk students?

   To answer this question, four training datasets, each representing a point of time at the semester, were used to train six different classifiers, four of which were single classifiers and the remaining two represented ensemble classifiers. Then, a comparison of the classification performance for all predictive models trained on each dataset was conducted. As detailed in Chapter four, the Extreme Gradient Boosting models consistently attained the highest performance across all datasets. Random Forest models were also able to achieve the second highest performance across all datasets. Both algorithms, Extreme Gradient Boosting and Random Forest, are based on ensemble methods. Hence, ensemble classifiers were found to achieve the best performance. Of the single classifiers, Artificial Neural Network models were able to achieve the best classification performance across

all dataset and were ranked at the third place with an AUC closer to the one achieved by the ensemble classifiers.

2) What accuracy can be obtained by using ML techniques based on data collected from the first weeks of a course?

   The classifiers trained on week zero's dataset, which represent data collected before the beginning of classes, achieved an AUC ranged from 89% to 83%, a sensitivity ranged from 64% to 47%, and a specificity ranged from 91% to 89%. Furthermore, Week four's classifiers achieved an AUC between 91% and 84%, a sensitivity between 66% and 57%, and a specificity between 92% and 90%. Additionally, Week eight's classifiers achieved an AUC of 91% to 84%, a sensitivity of 67% to 60%, and a specificity of 92% to 88%. Finally, Week 12's classifiers achieved an AUC from 92% to 85%, a sensitivity from 69% to 58%, and a specificity from 92% to 89%.

3) Can ensemble techniques improve the prediction accuracy of the base classifiers?

   The results of all experiments indicate that the performance of ensemble classifiers was superior to the performance of base classifiers. Chapter four provides more details on the performance of ensemble classifiers.

4) How does the prediction accuracy improve as the course progresses?

   The Extreme Gradient Boosting classifier, the top performing classifier across all datasets, was able to attain an AUC of ≈ 0.89 on week zero's dataset, and AUC of ≈ 0.90 on week four's dataset, an AUC of ≈ 0.91 on week eight's dataset, and an AUC of ≈ 0.92 on week 12's dataset. Additionally, the XGBoost classifier was able to achieve a sensitivity and specificity of ≈ 0.81, ≈ 0.82, ≈ 0.83, and ≈ 0.84

for week's zero, four, eight and 12 datasets, respectively. Hence, it can be concluded that the performance (i.e., AUC, sensitivity and specificity) of the XGBoost classifiers was able to improve by 1% for each subsequent dataset. However, this improvement is marginal in comparison to week zero's results. While the sample of this study was drawn from courses that made significant use of LMS, it was found that reliable predictions can be made based on week zero's model, which was trained with SIS data only. Consequently, week zero's model can be used to identify at-risk students even in courses that do not make significant use of LMS.

5) Can the predictive models achieve similar classification performance across on-campus and online courses?

   The results of the generalizability analysis detailed in Chapter four indicate that models trained with examples that include both on-campus and online students can achieve a similar performance. However, when trained on only one type of students' examples, such as on-campus students, similar results cannot be achieved when used to classify students of another type, such as online students. Additionally, the findings highlighted in question four's answer indicate the appropriateness of week zero's model to be used with all courses regardless of their LMS usage. Such findings demonstrate the generalizability of week's zero model.

6) What attributes are the best predictors?

   Table 37 shows the number of attributes used in each week's training and evaluation sets. These attributes were obtained from the SIS and LMS databases.

In the first section of Chapter four, variable importance plots and discussions were provided for each classifier. Overall, the attributes obtained from the SIS database demonstrated a much stronger predictive powers as evident by their domination of the top 15 most important variables plots for most of the classifiers.

Table 37. Number of Attributes in Each Week's Dataset

Week	Features Set	# of Features
Week 0	Full Set	46
	Filtered Set	35
Week 4	Full Set	98
	Filtered Set	35
Week 8	Full Set	239
	Filtered Set	120
Week 12	Full Set	427
	Filtered Set	150

Implications

The aim of this research was to address the challenges associated with incorporating predictive models that can effectively identify at-risk students into early warning systems. The challenges include the accuracy of timely predictions and the generalizability of predictive models across on-campus and online students. Results of this dissertation indicate the usefulness and effectiveness of ML techniques for early identification of at-risk students. It is hoped that the results of this study advance the understanding of the appropriateness and effectiveness of ML techniques when used for early identification of at-risk students. The results demonstrated the suitability and effectiveness of Extreme Gradient Boosting, Random Forest, and Neural Network classifiers for identifying at-risk students.

Recommendations

Based on the findings of this research, it is recommended to incorporate the suggested predictive models with the school's learning management system so that continuous predictions of at-risk students can be instantly delivered to students and instructors. Additionally, it would be useful to investigate another set of variables related to at-risk students, which can help in improving the performance of the classifiers. Furthermore, investigating the application of other ML techniques to the at-risk problem can identify another set of useful techniques. Finally, research can benefit from reproducing the results of this study in the context of another university and students' population.

Summary

This research focused on the development and evaluation of ML predictive models that can be used to effectively identify at-risk students. The identification of at-risk students needed to be early in the semester so that appropriate measures can be taken to help those struggling students succeed in their course of study. Early identification of at-risk students could help schools and instructors in providing at-risk students with a personalized and cost-effective remedial approach. To accomplish the goal of this study, four different datasets, each representing a point of time at the semester, were used to train and evaluate six different classifiers, four of which were single classifiers and the remaining two represented ensemble classifiers. The results indicate that ensemble classifiers were able to achieve the highest performance across all datasets. Furthermore, the Extreme Gradient Boosting models showed consistent and superior performance across all datasets. As a result, the XGBoost classifier was found to be the best

performing classifier suited for the at-risk students' classification problem. Additionally, the results of the generalizability analysis show that the models were able to attain a similar performance when used to classify on-campus and online students. Moreover, the results demonstrated the generalizability of week zero's model across all courses regardless of their LMS usage. Finally, the top performing classifiers achieved a satisfactory performance when trained with data that was available before the beginning of the semester and were able to improve by 1% for each subsequent four weeks dataset. While the improvement in the classification accuracy of weeks' four, eight, and 12 models was found marginal, week's zero model provided a satisfactory prediction that can be used to identify and help at-risk students even before the course starts.

References

ACT. (2015). *2015 Retention/Completion Summary Table*. Retrieved from http://www.act.org/content/dam/act/unsecured/documents/2015-Summary-Tables.pdf

Al-Radaideh, Q. A., Al-Shawakfa, E. M., & Al-Najjar, M. I. (2006). Mining student data using decision trees. In *International Arab Conference on Information Technology (ACIT'2006), Yarmouk University, Jordan.*

Arnold, K. E., & Pistilli, M. D. (2012). Course signals at Purdue: Using learning analytics to increase student success. In *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge,* 267-270.

Barber, R., & Sharkey, M. (2012). Course correction: Using analytics to predict course success. In *Proceedings of the 2nd international conference on learning analytics and knowledge,* 259–262.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*(Feb), 281-305.

Bork, R. H., & Rucks-Ahidiana, Z. (2013). Role ambiguity in online courses: An analysis of student and instructor expectations. *Retrieved from doi: http://dx. doi. org/10.7916/D8C24TGV*.

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Breiman, L. (1996). Bagging predictors. *Machine learning*, *24*(2), 123-140.

Calvo-Flores, M. D., Galindo, E. G., Jiménez, M. P., & Piñeiro, O. P. (2006). Predicting students' marks from Moodle logs using neural network models. *Current Developments in Technology-Assisted Education*, *1*, 586–590.

Chatterjee, S. (2016). *fastAdaboost: A Fast Implementation of Adaboost*. Retrieved from https://CRAN.R-project.org/package=fastAdaboost

Chen, T., He, T., Benesty, M., Khotilovich, V., & Tang, Y. (2018). *xgboost: Extreme Gradient Boosting*. Retrieved from https://CRAN.R-project.org/package=xgboost

DesJardins, S. L., Ahlburg, D. A., & McCall, B. P. (1999). An event history model of student departure. *Economics of Education Review*, *18*(3), 375-390.

Dietterich, T. G. (2000). Ensemble methods in machine learning. *Multiple classifier systems*, *1857*, 1-15.

Dietz-Uhler, B., & Hurn, J. E. (2013). Using learning analytics to predict (and improve) student success: A faculty perspective. *Journal of Interactive Online Learning*, *12*(1), 17-26.

Etchells, T. A., Nebot, À., Vellido, A., Lisboa, P. J. G., & Mugica, F. (2006). Learning what is important: Feature selection and rule extraction in a virtual course. In *ESANN,* 401–406.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, *55*(1), 119-139.

Gašević, D., Dawson, S., Rogers, T., & Gasevic, D. (2016). Learning analytics should not promote one size fits all: The effects of instructional conditions in predicting academic success. *The Internet and Higher Education*, *28*, 68-84.

Ginder, S.A., Kelly-Reid, J.E., and Mann, F.B. (2016). *Graduation rates for selected cohorts, 2007–12; Student financial aid, academic year 2014–15; Admissions in postsecondary institutions, Fall 2015: First look (Provisional Data)* (NCES 2017-084). U.S. Department of Education. Washington, DC: National Center for Education Statistics. Retrieved from http://nces.ed.gov/pubsearch.

Harvey, A., & Luckman, M. (2014). Beyond demographics: Predicting student attrition within the Bachelor of Arts degree. *The International Journal of the First Year in Higher Education*, *5*(1).

Hosmer, D. W. Jr, Lemeshow, S., & Sturdivant, R. X. (2013). *Applied logistic regression* (Vol. 398). John Wiley & Sons.

Hu, Y. H., Lo, C. L., & Shih, S. P. (2014). Developing early warning systems to predict students' online learning performance. *Computers in Human Behavior*, *36*, 469-478.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (Vol. 112). New York: springer.

Jayaprakash, S. M., Moody, E. W., Lauría, E. J., Regan, J. R., & Baron, J. D. (2014). Early alert of academically at-risk students: An open source analytics initiative. *Journal of Learning Analytics*, *1*(1), 6-47.

Kahu, E. R., Stephens, C., Leach, L., & Zepke, N. (2013). The engagement of mature distance students. *Higher Education Research & Development*, *32*(5), 791-804.

Kantardzic, M. (2011). *Data mining: concepts, models, methods, and algorithms*. John Wiley & Sons.

Karatzoglou, A., Smola, A., Hornik, K., & Zeileis, A. (2004). kernlab - An S4 Package for Kernel Methods in R. *Journal of Statistical Software*, *11*(9), 1–20. Retrieved from http://www.jstatsoft.org/v11/i09/

Kotsiantis, S. B., Pierrakeas, C. J., & Pintelas, P. E. (2003). Preventing student dropout in distance learning using machine learning techniques. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems,* 267–274.

Kotsiantis, S., Pierrakeas, C., & Pintelas, P. (2004). Predicting students' performance in distance learning using machine learning techniques. *Applied Artificial Intelligence*, *18*(5), 411–426.

Kovacic, Z. J. (2012). Predicting student success by mining enrolment data. *Research in Higher Education Journal*, *15*, 1.

Kuhn, M., & Johnson, K. (2013). *Applied predictive modeling*(Vol. 26). New York: Springer.

Kuhn, M. (2017). *CARET: Classification and regression training*. Retrieved from https://CRAN.R-project.org/package=caret

Kuhn, M., & Quinlan, R. (2017). *C50: C5.0 Decision Trees and Rule-Based Models*. Retrieved from https://CRAN.R-project.org/package=C50

Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. *R news*, *2*(3), 18-22.

Lizzio, A., & Wilson, K. (2013). Early intervention to support the academic recovery of first-year students at risk of non-continuation. *Innovations in Education and Teaching International*, *50*(2), 109-120.

Lykourentzou, I., Giannoukos, I., Mpardis, G., Nikolopoulos, V., & Loumos, V. (2009). Early and dynamic student achievement prediction in e-learning courses using neural networks. *Journal of the American Society for Information Science and Technology*, *60*(2), 372–380.

Macfadyen, L. P., & Dawson, S. (2010). Mining LMS data to develop an "early warning system" for educators: A proof of concept. *Computers & Education*, *54*(2), 588-599.

Meier, Y., Xu, J., Atan, O., & van der Schaar, M. (2016). Predicting grades. *IEEE Transactions on Signal Processing*, *64*(4), 959-972.

Minaei-Bidgoli, B., Kashy, D. A., Kortemeyer, G., & Punch, W. F. (2003). Predicting student performance: An application of data mining methods with an educational web-based system. In *Frontiers in education, 2003.*

Murtaugh, P. A., Burns, L. D., & Schuster, J. (1999). Predicting the retention of university students. *Research in Higher Education*, *40*(3), 355-371.

Niemi, D., & Gitin, E. (2012). *Using big data to predict student dropouts: Technology affordances for research.* International Association for Development of the Information Society.

Pascarella, E. T., & Terenzini, P. T. (2005). *How college affects students* (Vol. 2). K. A. Feldman (Ed.). San Francisco, CA: Jossey-Bass.

Peña-Ayala, A. (2014). Educational data mining: A survey and a data mining-based analysis of recent works. *Expert Systems with Applications*, *41*(4), 1432-1462.

Quinlan, J. R. (1996). Improved use of continuous attributes in C4. 5. *Journal of artificial intelligence research*, *4*, 77-90.

Quinn, F., & Stein, S. (2013). Relationships between learning approaches and outcomes of students studying a first-year biology topic on-campus and by distance. *Higher Education Research & Development*, *32*(4), 617-631.

R Core Team. (2017). *R: A language and environment for statistical computing*. Vienna, Austria. Retrieved from https://www.R-project.org/

Ridgeway, G., & Others. (2017). *gbm: Generalized Boosted Regression Models*. Retrieved from https://CRAN.R-project.org/package=gbm

Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review*, *33*(1), 1-39.

Romero, C., & Ventura, S. (2013). Data mining in education. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *3*(1), 12-27.

Romero, C., López, M. I., Luna, J. M., & Ventura, S. (2013). Predicting students' final performance from participation in on-line discussion forums. *Computers & Education*, *68*, 458-472.

RuleQuest Research. (2017). *Is C5.0 Better Than C4.5?* Retrieved from http://rulequest.com/see5-comparison.html

Russell, S., & Norvig, P. (2010). Artificial intelligence: a modern approach. *Prentice Hall Press, Upper Saddle River, NJ, USA.*

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, *3*(3), 210-229.

Sander, P. (2016). *Using learning analytics to predict academic outcomes of first-year students in higher education*. Retrieved from https://scholarsbank.uoregon.edu/xmlui/handle/1794/21969

Schneider, M., & Yin, L. (2011). *The high cost of low graduation rates: How much does dropping out of college really cost?*. American Institutes for Research.

Seidman, A. (2012). *College student retention: Formula for student Success (2nd ed.)*. Lanham, MD: Rowman & Littlefield.

Smith, V. C., Lange, A., & Huston, D. R. (2012). Predictive modeling to forecast student outcomes and drive effective interventions in online community college courses. *Journal of Asynchronous Learning Networks*, *16*(3), 51–61.

Thai-Nghe, N., Busche, A., & Schmidt-Thieme, L. (2009). Improving academic performance prediction by dealing with class imbalance. In *The Ninth International Conference on Intelligent Systems Design and Applications, 2009. ISDA'09.* 878–883.

Tinto, V. (1987). *Leaving college: Rethinking the causes and cures of student attrition*. University of Chicago Press.

Tinto, V. (2012). *Completing college: Rethinking institutional action*. University of Chicago Press.

U.S. Bureau of Labor Statistics. (2016). *Earnings and unemployment rates by educational attainment*. Retrieved from https://www.bls.gov/emp/ep_table_001.htm

Venables, W. N., & Ripley, B. D. (2002). *Modern Applied Statistics with S* (Fourth Edition). New York: Springer.

Weihs, C., Ligges, U., Luebke, K., & Raabe, N. (2005). klaR Analyzing German Business Cycles. In D. Baier, R. Decker, & L. Schmidt-Thieme (Eds.), *Data Analysis and Decision Support* (pp. 335–343). Berlin: Springer-Verlag.

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Zafra, A., Romero, C., & Ventura, S. (2011). Multiple instance learning for classifying students in learning management systems. *Expert systems with applications*, *38*(12), 15020–15031.

Zafra, A., & Ventura, S. (2009). Predicting student grades in learning management systems with multiple instance genetic programming. *International Working Group on Educational Data Mining*.

Zhang, H. (2004). The Optimality of Naive Bayes. In *FLAIRS Conference* (pp. 562-567).

Zhang, Y., Fei, Q., Quddus, M., & Davis, C. (2014). An examination of the impact of early intervention on learning outcomes of at-risk students. *Research in Higher Education Journal*, *26*, 1.